Clean Up your Photo Metadata

LINUX PRO MAGAZINE

ISSUE 256 – MARCH 2022

Facial Recognition

Authenticate with a glance

Host-INT: Monitor network traffic from the data plane

Smart Contracts: Tricks with the Ethereum blockchain

Light Painting with a Raspberry Pi

PhotoPrism: Use AI to organize your photo collection

FOSSPicks DISCOVER THE OBSIDIAN KNOWLEDGE EDITOR

LINUX NEW MEDIA The Pulse of Open Source

Business Desktop Replacement

TUXEDO InfinityBook S 17

Intel Core i7-11370H
Intel Iris Xe Graphics

17'' display. 15'' format.
93 % Screen-to-body ratio

Lift-up hinge
Enhanced ergonomics & cooling

73 Wh battery
up to 16 h runtime

100%
Linux

5
Year
Warranty

Lifetime
Support

Built in
Germany

German
Privacy

Local
Support

TUXEDO COMPUTERS

tuxedocomputers.com

TO THE COLORS

Dear Reader,

Users of the *colors.js* color conversion library were surprised recently when their printers started to print "Liberty Liberty Liberty…" and then began spewing random characters in an infinite loop. The weird appearance of the printed page made those users wonder if a vandal had broken into the source code, but the truth that began to emerge was much stranger.

It turns out that the developer of the *colors.js* and *faker.js* libraries sabotaged the code himself. Marak Squires had apparently become disillusioned with the way large companies were using his code without contributing or providing compensation. On November 2020, he wrote on the *faker.js* GitHub page, "Respectfully, I am no longer going to support Fortune 500s (and other smaller-sized companies) with my free work. There isn't much else to say….Take this as an opportunity to send me a six figure yearly contract or fork the project and have someone else work on it."

At the time, it wasn't clear how the story would end. This month, we all found out.

Although *colors.js* and *faker.js* are independent projects that are largely overseen by a single volunteer, they have quite a large footprint in the world. According to reports, *colors.js* receives more than 22 million downloads per week and is integrated into many open source tools by commercial vendors, including the Amazon Cloud Development kit.

This kind of story always makes us want to choose sides for a proverbial "haves and have nots" or "business" versus "everyone else" debate. However, it would not be accurate to assume that the Free Software founders opposed commercial uses. The four freedoms that underlie the Free Software movement tolerate no restrictions on how the code will be used (as long as it stays free), and the commentaries at GNU.org make it clear that prohibiting commercial use is just another limitation on freedom.

One of the reasons the early Free Software pioneers didn't care if someone else got rich was because they totally didn't believe that getting rich was all that important. They wanted an identity for Free Software that was independent of wealth, and to them, the best way to ensure that was not to restrict wealth but to not care about it. Of course, they were young back then. As volunteers get older, they start to think about a mortgage and college tuition for their kids, and the idea of contributing for the pure joy of it while a large corporation is deriving value from the work must be quite unsettling.

I can see why this developer was ready to stop giving his work away for free – especially with mega vendors like Amazon using his code and not pitching in or compensating him. However, the problem with this kind of dramatic exit is that it plays into the hands of those who depict Free Software as a playground for unruly anarchists who can't be trusted with real business. I can imagine there are Microsoft sales reps out there right now scheming on how to work this episode into their pitch.

Note that none of this has anything to do with Linux – at least directly. The affected projects are based on JavaScript and are therefore cross-platform, but in the long run, this kind of theatrical protest is not a good look for the open source development model, which means it isn't good for the Linux community.

Independent open source developers burn out all the time, and, ideally, resources are available within the project for an orderly transition. However, many projects are too small or too fragmented to provide the needed continuity. What is striking about this case is that Squires served notice all the way back in 2020 that, unless a paycheck was on the way, someone else would need to take over. Should we have been listening? Perhaps it is time for the Linux Foundation or another of the multimillion dollar organizations dedicated to overseeing the Linux image to provide some direct outreach and transition support for these small projects that give Linux so much of its luster. ■■■

Joe

Joe Casad,
Editor in Chief

LINUX MAGAZINE

MARCH 2022

ON THE COVER

TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!
SEE PAGE 6 FOR DETAILS

antiX 21 and Haiku R1/Beta 3

Two Terrific Distros on a Double-Sided DVD!

antiX 21

64-bit

Based on Debian stable, antiX 21 is designed to be light and efficient. The full installation version available on this month's DVD is 1.45GB and requires 256MB RAM. However, for older computers, antiX also has a 800MB base version, a 440MB core version, and a 180MB net version. The core and the net versions only install from the command line and without encryption. Both are recommended for those who want total control over the installation, which is necessary for a secure system. From within antiX, users can also set up a live flash drive from which to boot a computer. In addition, antiX is one of the rapidly diminishing number of distributions that still includes a 32-bit version.

One of antiX's standout features is its installer, which gives concise, clear help at each step of the process. The installer is notable for its unobtrusive security features, including a choice between a legacy kernel, which is more secure, and a modern kernel which might have more bugs, as well as a small graph that indicates password strength. Once installed, antiX offers a desktop with a system summary widget and a menu with prominently displayed administrative tools that include numerous command-line tools.

Don't let antiX's drab default wallpaper deceive you: It can be easily changed. Users of any level can appreciate antiX's speed and minimalism, as well as its emphasis on hands-on computing.

Haiku R1/Beta 3

64-bit

In the late 1990s, BeOS was a proprietary operating system with cult followers. BeOS was among the first operating systems to use multiple processors and extensive multithreading. It was known for its speed, stability, and its focus on home users. BeOS failed commercially, but its spirit lives on in Haiku, a small open source project that began with BeOS source code.

Haiku has yet to reach general release, but with this version, it is stable enough for most purposes. It installs with its own apps plus a selection of KDE apps. Other standard apps like LibreOffice and Firefox are available in the repositories. Advanced users might be especially interested in Haiku's package management system, in which packages are block-compressed system images that are loaded at bootup. When uninstalled, packages are saved in a special folder, making rollbacks a matter of moving the uninstalled packages back to their original position.

For all users, installing Haiku is a chance to see an operating system in development. Installation is fast and simple, and Haiku's desktop includes several features that are unique among open source interfaces.

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

MEET 150+ LEADING CLOUD & TECHNOLOGY PROVIDERS.
JOIN HUNDREDS OF BUSINESS AND TECHNOLOGY LEADERS FOR THOUGHT-PROVOKING PANEL DISCUSSIONS, INSPIRATIONAL KEYNOTES AND PRACTICAL CASE STUDIES.
UNMISSABLE NETWORKING OPPORTUNITIES.
REGISTER FOR YOUR FREE TICKET TODAY:
WWW.CLOUDEXPOEUROPE.COM/ADMINMAGAZINE-LINUXMAGAZINE

Transform the future of your business.

After two-years out, the global event of choice for everyone committed to the design, build and management of digital initiatives and technology architecture is back. If you're a technologist or a business leader in the public, voluntary and private sector, this is the place to give your future a jet-propelled boost of inspiration, ideas and innovation. All the very best suppliers and providers combine with expert-delivered content in one unmissable event.

Learn about the As-A-Service Model, Digital Acceleration, Emerging Tech, Hybrid and Multi-cloud, Sustainable Cloud and more. Come and see your digital transformation soar.

Register for your FREE ticket today:
www.cloudexpoeurope.com/AdminMagazine-LinuxMagazine

CLOUD EXPO EUROPE

2 - 3 March 2022 ExCel, London
www.cloudexpoeurope.com

PART OF
TECH SHOW LONDON

INCORPORATING
CLOUD EXPO EUROPE | DEVOPS LIVE | CLOUD & CYBER SECURITY EXPO | BIG DATA & AI WORLD | DATA CENTRE WORLD

THE MOST IMPORTANT TECHNOLOGY EVENT FOR BUSINESS IN THE UK

ORGANISED BY
CloserStill

PLATINUM SPONSORS
wasabi, MariaDB, IONOS CLOUD

GOLD SPONSORS
LzLabs, aiven, APPVIA, cloudEQ

PARTNERS
Tech, electronic specifier, STARTUPS MAGAZINE, CIOINSIGHTS, EXCEL LONDON, Silicon.co.uk, CLOUD INDUSTRY FORUM, Computing Security, NETWORKcomputing, STORAGE MAGAZINE, TechTarget, techUK, ADMIN, elitebusiness, MEDIA7, OpenUK, LINUX MAGAZINE, CryptoNewsZ, Coinmercury, THE FINTECH TIMES, Packt, MARKETSANDMARKETS

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Linux Mint 20.3 Now Available

Users of the popular Linux Mint distribution can celebrate the new year by downloading a new release, version 20.3 (*https://linuxmint.com/download_all.php*). This latest iteration is based on Ubuntu 20.04.5 LTS. Although it doesn't have any game-changing new features, it does offer a lot of subtle UI tweaks and a very helpful document manager app.

As for the polish, the default Mint theme doesn't lean so much on the color green and includes larger titlebars, bigger controls, and rounded corners. A number of the default apps also default to a dark theme.

Some of the app improvements include the calendar tool showing events from multiple sources, some tweaks to the Nemo file manager (including the ability to rename a moved or copied file to avoid overwriting a duplicate), new styling for the run dialog, major updates to the Notes app (which includes a new search function), a channel search option for Hypnotic (Mint's internet TV app), and the ability to switch between tabs with the Ctrl+Tab keyboard shortcut with the Xed text editor.

There's also the new Thingy app, which is a document manager that provides quick access to favorite and frequently opened documents. Thingy also keeps track of reading progress, so when you re-open a document it'll start where you left off.

To find out everything that's in the new version, make sure to check out the Linux Mint 20.3 release notes (*https://www.linuxmint.com/rel_una_cinnamon.phpd).*

Linux Gets an Exciting New Firmware Feature

When you upgrade your motherboard firmware (such as the BIOS or UEFI), you have to reboot your system. Thanks to a new patch from Intel, both BIOS and UEFI updates can be done without forcing a reboot.

How is this possible? Currently, an upgrade is done by uploading the firmware from within the operating system. The desktop or server is then rebooted, at which point the firmware is transferred to the motherboard and is flashed to either the BIOS or UEFI. However, there's a new API specification, called Platform Firmware Runtime Update and Telemetry (PFRUT), which makes it possible to flash the firmware without the reboot. Intel has been working on PFRUT (previously dubbed Seamless Update) for quite a while now, in order to reduce downtime for servers. The idea is to enable such machines to reach that mythical 100 percent uptime.

The new driver, `pfr_update`, will be introduced in Linux Kernel 5.17 and is designed primarily for system firmware updates to patch critical bugs and security issues. This would make it possible for admins to patch firmware for critical issues, without having to suffer downtime.

One of the biggest surprises to come along with PFRUT is that it will only be available for Linux (so Windows users need not apply).

You can read about the patch in this Kernel.org entry (*https://git.kernel.org/pub/scm/linux/kernel/git/rafael/linux-pm.git/commit/?h=linux-next&id=0db89fa243e5edc5de38c88b369e4c3755c5fb74*).

elementary OS 6.1 Has Been Released

The developers of elementary OS have been hard at work delivering the first point release for Odin (the sixth iteration of the distribution). Normally a point release wouldn't receive a new name and identity, but the developers felt there was enough polish added to warrant the change.

The new release of elementary OS comes with an AppCenter that continues to fill out with applications. Since elementary OS (*https://elementary.io/*) has added Flatpak support, you'll find over 90 curated apps in the AppCenter. The shift from Debian packages to Flatpak has made it possible for developers to push out rapid and frequent updates. And thanks to the added Flatpak support, along with curated apps you'll find plenty of non-curated apps to fill in the gaps. The AppCenter itself has received plenty of attention with a reworked home page and banners featuring the most recently released applications.

As for the desktop, release 6.1 brings improvements and polish across the entire space. You'll find a redesigned quick window switcher (used with Alt+Tab), refreshed dialogs, an improved File Chose portal, a dark style that's more widely respected across desktops, a more powerful search within the Applications menu (which can now search for bookmarked folders and locations such as downloads, pictures, and network shares).

And for those who prefer to set their hostname during the installation, the elementary OS 6.1 installer now allows you to do just that.

There's plenty more updates and polishing that went on to create elementary OS 6.1. Read more in the official elementary OS blog (*https://blog.elementary.io/elementary-os-6-1-available-now/*).

Intel Releases Linux Patch for Alder Lake Thread Director

The Performance and Efficiency cores within Intel's Adler Lake CPUs have received patches to dramatically increase performance with the Linux operating system.

Soon after Microsoft released Windows 11, it became clear that the Linux operating system lagged behind the competition in performance. The reason for this was because Linux lacked adequate support for Intel's Thread Director technology (created from the Enhanced Hardware Feedback Interface), which grants proper access to the high-performance Golden Cove cores and the energy-efficient Gracemont cores.

The current firmware for Linux relies on an algorithm to plan which P/E cores are utilized by the ITMT/Turbo Boost Max 3.0 driver. That method is not nearly as efficient as Intel's new patch. The company explains the patch by saying:

"The Intel Hardware Feedback Interface (HIFI) provides information about the performance and energy efficiency of each CPU in the system. It uses a table that is shared between hardware and the operating system. The contents of the table may be updated as a result of changes in the operating conditions of the system (e.g., reaching a

MORE ONLINE

Linux Magazine
www.linux-magazine.com

ADMIN HPC
http://www.admin-magazine.com/HPC/

Parallel and Encrypted Compression
• Jeff Layton
Compression tools are underutilized. I base this statement on the observation that data, once created, is seldom accessed again. To me, this seems like a great opportunity to use compression tools to reduce the used storage capacity and perhaps save a little money.

ADMIN Online
http://www.admin-magazine.com/

Run applications in a containerized sandbox with Firejail
• Matthias Wübbeling
Isolate popular applications in flexible, easy-to-set-up, and easy-to-take-down containers with Firejail.

Nested Kubernetes with Loft
• Martin Loschwitz
Kubernetes has limited support for multitenancy, so many admins prefer to build multiple standalone Kubernetes clusters that eat up resources and complicate management. As a solution, Loft launches any number of clusters within the same control plane.

Linux apps on Windows 10 and Chrome OS
• Christian Knermann
Microsoft and Google have upgraded their in-house operating systems, Windows 10 and Chrome OS, with subsystems to run Linux. We look into their highly different approaches.

thermal limit) or the action of external factors (e.g., changes in the thermal design power)."

The HIFI calculates the power efficiency and performance of the CPU, gives the core a numerical value, and communicates that information to the operating system.

This new set of patches is still in the revision stage and there has yet to be an announcement as to when they will be made available to the kernel (or if they'll make it into version 5.17). Read more about this update on *https://lore.kernel.org/lkml/20211220151438.1196-1-ricardo.neri-calderon@linux.intel.com/*.

New Multiplatform Backdoor Malware Targets Linux, macOS, and Windows

The first signs of SysJoker appeared in December 2021, when researchers at Intezer were investigating an attack on a Linux web server. This malware is written in C++ and each variant is specifically tailored for the operating system it attacks. VirusTotal was unable to detect the malware, even using 57 different detection engines.

Once the malware has been deployed, it fetches the SysJoker zip file from GitHub, unpacks it, and executes the payload. The payload gathers information about the machine, stores and encodes the results in a JSON object, creates persistence, reaches out to a C2 server (using a hard-coded Google Drive link, where the server is instructed to install additional malware), and runs commands on the infected device.

Intezer has provided a list of indicators for SysJoker for each operating system. On Linux, the files and subdirectories are created under `/.Library/` and persistence is created with the cron job `@reboot (/.Library/SystemServices/updateSystem)`. If you discover such a cron job, it's imperative that you kill all related processes, manually delete the files and cron job, scan the system to ensure all malicious files have been removed, and check for any weakness that might have allowed the attackers access to your server.

Find out more about SysJoker in the original Intezer report (*https://www.intezer.com/blog/malware-analysis/new-backdoor-sysjoker/*).

WhiteSource Releases Free Log4j Detection Tool

As the Log4j vulnerability continues to wreak havoc on the IT landscape, everyone is trying to prevent disaster from striking. A number of companies and development teams have released tools to help with the detection and remediation of the vulnerability. One such company is WhiteSource. Their new tool, Log4j Detect (*https://github.com/whitesource/log4j-detect-distribution*), is an open source command-line utility that scans your projects to detect the following known CVEs:

- CVE-2021-45046
- CVE-2021-44228
- CVE-2021-4104
- CVE-2021-45105

Once the scan is complete, it will report back the exact path of the vulnerable files as well as the fixed version you'll need to remediate the issue. Log4j Detect should be run within the root directory of your projects and will also search for vulnerable files with both the `.jar` and `.gem` extensions. Log4j Detect supports the Gradle, Maven, and Bundler package managers.

In order for Log4j Detect to run properly, you'll need to install either `gradle` (if the project is a Gradle project) or `mvn` (if the project is a Maven project). The developers have also indicated both Maven and Bundler projects must be built before scanning. Once you have Log4j Detect installed, the scan can be issued with the command `log4j-detect scan -d PROJECT` (where `PROJECT` is the directory housing your project).

For more information about this tool, make sure to read through the project README (*https://github.com/whitesource/log4j-detect-distribution/blob/main/README*).md).

Zack's Kernel News

Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

A Butterfly Flaps Its Wings

Andy Shevchenko, who is not a butterfly, happened to remark one day on the Linux Kernel Mailing List: "would it make sense to enable -Werror for default warning level, let's say W = 0, at some point?" And this ended up producing great storms of change across the world.

The `-Werror` option is a command-line option for the GNU C Compiler (GCC). When compiling, GCC will detect many errors in your code, report them to you, and abort the attempt to compile your code. Perhaps you haven't quite made a real error, but your code is still not a good usage of the C language. In that case, GCC produces a warning instead and continues chugging away to build your executable. However, maybe you are so enthusiastic about proper C usage that you want GCC to treat those bad usages as if they were errors. The `-Werror` option tells GCC to do that.

Masahiro Yamada pointed out to Andy that "Every GCC release adds new warning options." And that "Enabling -Werror by default means the kernel build is suddenly broken with new compilers."

Andy replied that in spite of Masahiro's objection, using `-Werror` would at least "keep our hand on the pulse of the changes, right?"

Nick Desaulniers from Google pointed out that "Google's pixel kernel team carries an out of tree patch creating exactly such a config. It helps them keep their kernels building warning free."

So there was at least a precedent for hoping such a thing could be done. But Nick himself was opposed to the idea in spite of it being standard practice at his own company. He said, "if a build isn't warning free, it can be difficult to get there; you need to at least disable the config to see how many warnings you have, and identify which are lower hanging fruit. It also makes compiler upgrades excessively difficult. In my experience on Android, if folks are too busy to address compiler warnings, then new warnings added by a new compiler version just get turned off and never addressed. My experience with the kernel has been that fixes for different warnings also take varying amounts of time to get accepted and work their way through mainline, meanwhile builds are broken."

Linus Torvalds also responded to Andy's original post regarding `-Werror`, saying:

"I'd love to, and would have done that a long time ago, but we just haven't been able to depend on the compilers not having random warnings in random versions.

"And making it a config option doesn't work well either, simply because that will just mean that bots will randomly fail if they set that option, and if we make it harder to set developers won't have it set anyway, and it doesn't help much.

"End result: using -Werror works wonderfully in controlled environments. Not so wonderfully in the general random mess.

"That said, I've been so aggressive against accepting new warnings that I've considered this unconditionally anyway, and then dealing with compiler version fallout by just disabling certain compilers and/or certain options.

"But it would almost certainly be pretty painful."

Elsewhere, in an entirely different thread, Kees Cook posted a patch for Linus, which Linus rejected, saying, "You can't add new warnings without fixing them, and this adds some HORRENDOUSLY ugly new warnings that would most definitely hide other warnings."

In a subsequent post, Linus added, "I really want to enable -Werror at some point, but every time I think I should, I just end up worrying about another random new compiler (or a random old one). We do have -Werror in various configurations (and in some sub-trees)."

But then, fuming over his own email, Linus responded to himself, saying:

"Whatever. I'll just make a new config option, make it 'default y', and it will be on for anybody doing allmodconfig builds etc.

Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

"And if people have new compilers, or odd configurations that still cause warnings, they can turn it off, but hopefully this will make compiler warnings in linux-next (or any other automated builds) cause a lot more noise."

In other words, kaboom. Linus decided to enable `-Werror` and just deal with the enormous fallout of kernel builds breaking across the known universe.

In yet a third entirely different thread, the kaboom-heard-round-the-world echoed in the ears of Nick (the Google Android guy) who had already objected to the kabooming act itself. He saw that Linus had already committed a patch enabling `-Werror`, and he began to wail into the night. He said, "While I can appreciate the intent of enabling -Werror, I don't think it is the right tool to address the root cause of developers not testing certain toolchains or configurations, or taking existing reports they're getting serious enough."

He added, "I'd also like to see such a patch sent formally to the list for discussion and have time to soak in next rather than be merged directly into mainline without either."

He said that `-Werror` might be OK if all existing warnings had already been fixed or if the tool chain would never be updated. But Nick said, "Unfortunately, none of the above is the case for the Linux kernel at this time."

Nick added that at least at Google "This change has caused nearly all of our CI to go red, and requires us to now disable CONFIG_WERROR until every last target and every last config is addressed. Rather than require everyone to disable the above config to keep builds going, perhaps certain CI systems should instead set CFLAGS_KERNEL = -Werror."

But Linus put his foot down once and for all and said:

"It was merged in response of _years_ of pain, with the last one just being the final drop.

"I'm not going to revert that change. I probably will have to limit it (by making that WERROR option depend on certain expectations), but basically any maintainer who has code that causes warnings should expect that they will have to fix those warnings.

"If it's clang that generates bogus warnings, then we'll have to start disable clang warnings. The clang people tend to be proud of their fewer false positives, but so far looking at things, I am not convinced.

"And I'm most definitely not convinced when the 'let's finally enable -Werror after years of talking about it', people end up going 'but but but I have thousands of warnings'.

"That's the POINT of that commit. That 'but but but I have thousands of warnings' is not acceptable.

"I spent hours yesterday getting rid of some warnings. It shouldn't be on me fixing peoples code. It shouldn't be on me noticing that people send me crap that warns.

"And it really shouldn't be 'Linus cares about warnings, so configurations that Linus doesn't test can continue for years to have them'.

*"My 'no warnings' policy isn't exactly new, and people shouldn't be shocked when I then say 'time to clean up *YOUR* house too'."*

Nick pointed out that this patch was merged without any public discussion over a holiday weekend. It gave no one a chance to test it relative to their various configurations and tool chains. He added to Linus, "Any given maintainer sending you a PR cannot (and should not, IMO) know under what combination of configs, targets, and toolchains you'll test under, and -Werror isn't going to help them figure it out."

He went on, "To be clear, you have merged patches into mainline that broke the build for combinations of configs/targets/toolchains that you are not testing. It's not realistic for you or any one person to test all such combinations either."

And Nick concluded, "Fixing warnings in the Linux kernel for all possible configs, targets, and toolchains while the toolchains continue to add more diagnostics is more sisyphean than digging a hole in wet sand."

Fangrui Song also pointed out that "Default WERROR makes building old kernels with new compilers more painful."

Marco Elver submitted a one-line patch, changing the `-Werror` option from on by default to on for test builds. He said that this "might move this new Kconfig option towards its appropriate usecases by default. The intent is not to dispute the usefulness of -Werror, but select the appropriate usecases by default, limiting friction for all those who can do little more than say CONFIG_WERROR = n."

Guenter Roeck, Nathan Chancellor, and Randy Dunlap liked Marco's patch, and Linus accepted it, saying "That seems reasonable. It very much is about build-testing."

Mark Brown, however, said, "having -Werror on by default for defconfigs is going to cause issues for bisection and just generally noticing promptly when runtime issues are introduced."

Steven Rostedt, deviating from the general mood of nauseous shock pervading the mailing list, didn't really object to `-Werror`. He said, "ktest has a way to create a list of current warnings, and then test your code against it, and will fail on any new warning. I run this on all my pull requests to make sure that I do not introduce any new warnings. That said, I still get bug reports on various configs that I did not test, where my code introduces a warning. I hate to be the one that fails their builds."

Pavel Machek did not like `-Werror`, saying that it would affect the stable kernel tree. The purpose of the stable tree is to accept only bug fixes. Something like `-Werror` would suddenly inspire a lot of people to try to fix all the newly appearing bugs. This, he said, would result in a lot of patch submissions to the stable tree, which could introduce more instability than it fixed.

On that same topic, Greg Kroah-Hartman said, "I will not be backporting this ['-Werror'] patch to older stable kernels, but I _want_ to see stable builds build with no warnings. When we add warnings, they are almost always things we need to fix up properly. Over time, I have worked to reduce the number of build warnings in older stable kernels. For newer versions of gcc, sometimes that is impossible, but we are close…."

Pavel replied, "You clearly can't backport this patch, but for 5.16-stable, you'll have it in, and now warnings are same as errors… and I don't believe that's [a] good idea for stable."

But Greg disagreed. He felt it was a fine idea for the stable tree, and that "it will force us to keep these trees clean over time." He also added that in the "worst case, we disable it in 4 years when gcc 15 or so generates so many errors we can't resolve them in this old kernel."

Florian Weimer raised some concerns about `-Werror` in general, saying "there are also warnings which are emitted by the GCC middle-end (the optimizers), and turning on -Werror for those is very problematic. These warnings are very target-specific and also depend on compiler version and optimization parameters." And he added, "GCC also lacks a facility to suppress warnings if they concern code that was introduced during optimization and removed again later (e.g. inlining, constant propagation, dead code removal)."

Linus jumped back into the conversation here, regarding Florian's statement about "middle-end" warnings. Linus said to him:

"People say that, but let's face it, that's simply not true.

"There are real problematic warnings, and we just turn those warnings off. People who want the self-flagellation can enable them with W = 1 (or bigger values), and spend their life fighting stupid random compiler warnings that have tons of false positives.

"But the fact is, I've required a warning-free build on x86-64 for anything I notice for the last several years by now, and it really hasn't been a problem.

"What _has_ been a problem is that (a) build bots don't care about [warnings] and (b) the configs I don't personally test (other non-x86-64 architectures stand out, but there's certainly been other configuration issues too).

*"But 'bogus compiler warnings' is very much *not* in that list of problems.*

"I've looked at a lot of the warnings that are now errors, and while a number of them have made me go 'So why didn't we see that on x86-64?' not one of them has actually made me go '-Werror was wrong'.

"Because EVERY single one I've seen has been for something that should have been fixed. Presumably long long ago, but the warning it generated had been ignored.

"So stop with the 'some warnings just happen' crap. Outside of actual compiler bugs, and truly stupid warnings (that we turn off), that's simply not true.

"And yes, those compiler bugs happen. The new warning already found one issue with current gcc trunk (non-released). So right now the count is 'lots of valid warnings, and one compiler bug that was found _thanks_ to me enabling -Werror'.

"Yes, we have issues with having to work around older compiler bugs. Those aren't going away, and yes, -Werror may well mean that non-x86-64 people now have to deal with them.

"And yes, this is painful. I'm very much aware of that. But we just need to do it. Because the warnings don't go away on their own, and not making them fatal clearly just means that they'll stay around forever."

Thus spake Linus, and that was the end of that. At least for that thread in the mailing list.

Later, Linus gave an update on how the kernel was handling `-Werror` so far. He said:

"I've spent a fair amount of this week trying to sort out all the odd warnings, and I want to particularly thank Guenter Roeck for his work on tracking where the build failures due to -Werror come from.

"Is it done? No. But on the whole I'm feeling fairly good about this all, even if it has meant that I've been looking at some really odd and grotty code. Who knew I'd still worry about some odd EISA driver on alpha, after all these years? A slight change of pace ;).

"The most annoying thing is probably the 'fix one odd corner case, three others rear their ugly heads'. But I remain convinced that it's all for a good cause, and that we really do want to have a clean build even for the crazy odd cases.

"We'll get there."

Still later and in an unrelated thread, someone complained about the deep, echoing silence from Linus in response to a patch. But Linus explained, "Normally I get to clean up my inbox the week after the merge window, but the -Werror things kept my attention for one extra week, and so my mailbox has been a disaster area as a result. So only today does my inbox start to look reasonable again after the merge window."

A month later, Linus announced Linux Kernel 5.15, with the `-Werror` patch attached. He remarked, "This release may have started out with some -Werror pain, but it calmed down fairly quickly and on the whole 5.15 was fair[ly] small and calm. Let's hope for more of the same – without Werror issues this time – for the upcoming merge window."

And that's the story.

Kaaa…boom? ■■■

Facial authentication with Howdy

Howdy, Friend

Howdy brings the convenience of facial authentication to Linux.

By Karsten Günther

Futurists and entrepreneurs have long been obsessed with the idea of biometric authentication. Every body is unique – could the unique features of your anatomy be used to prove that you are you? Police departments have used fingerprints to identify suspects for years, and several computer systems now allow fingerprint authentication. But for many researchers, the holy grail for biometric authentication has always been facial recognition.

The human brain authenticates humans through facial recognition all the time: That is what happens when you "recognize" someone. Of course, when a camera takes a picture of a person, it is most likely a picture of a face. Your face is looking at your laptop or phone all the time. What could be more convenient than just training your electronic device to look back at you and determine if the face is familiar?

Windows Hello technology, which is built into Microsoft systems, gives Windows users an option for facial identification. The Linux community can now access facial identification through an open source application known as Howdy [1]. Lem Severein from the Netherlands has been developing Howdy for more than four years, and the software is currently available on GitHub.

As you can probably guess, facial authentication system requires two important components:

- Imaging – a way to capture an image of the person who is trying to authenticate
- Artificial intelligence – a way to compare that image with a reference image of the user

Once the identity is confirmed, the application then needs a way to pass the approval on the operating system.

Howdy uses the OpenCV computer vision library [2] for imaging and facial recognition. Fortunately for the Linux community, the Pluggable Authentication Module (PAM) system built into Linux makes it easy to integrate new authentication technologies (see the box entitled "PAM Authentication"). Howdy uses the OpenCV vision library for image processing and facial recognition; then it passes control on to PAM.

Severein points out that Howdy is not intended as a complete security system by itself. As the website states, it is "... in no way as secure as a password and will never be.... Howdy is a more quick and convenient way of logging in, not a more secure one."

One common use for facial authentication tools is to re-authenticate a user after the user steps away from the computer. Howdy can serve as a first line of defense in casual settings, but be aware that this technology is still experimental – if you are counting on enterprise-level security, better to stick with more conventional approaches.

The Camera

It takes a camera to see a face. Today, IR (near field infrared) cameras, which are not normally installed in laptops as standard equipment, are a popular option for reliable facial recognition. Howdy theoretically works on laptops that only have a standard webcam, but it won't be as secure or as reliable. The GitHub project `linux-enable-ir-emitter` [3], which is designed for use with Howdy, offers support for infrared cameras that are not directly enabled out-of-the box.

Howdy

Howdy has not yet reached the package sources for many Linux distributions; however, the developer provides instructions for setting up the software on popular Linux systems. In particular, Howdy offers strong support for Ubuntu, offering a separate PPA for Ubuntu systems. For Arch Linux, the required packages are in the AUR, with instructions in the ArchWiki [4].

PAM Authentication

The PAM system built into Linux offers a way to integrate low-level authentication processes into higher-level APIs. PAM consists of a cascade of modules that control authentication. On one hand, this means that you can apply several procedures in succession: for example, first facial recognition and then, if this fails (or in addition), password input. On the other hand, you can also use PAM to provide different procedures for different programs: For instance, use facial recognition to unlock the desktop, but require a password to install new programs.

As a Python program, Howdy has a number of distribution-specific dependencies, such as *python-pam*. In addition to using OpenCV for face recognition, Howdy leverages the Hierarchical Data Format (HDF5) for data storage and FFmpeg for reading the video stream.

The basic installation is quite simple on most distributions: Install the *howdy* package via your package manager. The system drags in all the dependencies it needs. With Arch Linux, you might have to add some packages by hand, such as the *python-pam* package.

Try It Out

Howdy requires some manual configuration work and face recognition training. The basic software configuration and training are handled by the `howdy` command in the terminal, in combination with a few subcommands (as shown in Table 1). You can also use options to simplify the steps for some actions (see Table 2).

To set up the software, use:

```
sudo howdy config
```

Table 1: Subcommands

Parameter	Function
`add`	Add profile (model) for current user
`test`	Test facial recognition
`clear`	Delete profile
`config`	Create or edit configuration
`disable`	Disable Howdy
`list`	Display profiles
`remove ID`	Delete profile with `ID`
`snapshot`	Create snapshot
`version`	Display version

Table 2: Options

Option	Function
`-h`	Online help
`-y`	Skip profile name query
`-U user`	Define profile for `user`

This process will start the default editor with a fairly compact configuration file that documents all settings in detail. The few changes necessary for some initial experiments are shown in Listing 1.

By default, the `detection_notice` parameter is set to `false`, which prevents Howdy from reporting successful face recognition. For initial testing, however, this feature is very useful, which is why you will want to specify a value of `true` instead (line 4).

By far the most important specification refers to the video device used (line 9). The IR camera was available as `/dev/video2` in our lab; normal webcams are often designated `/dev/video0` or `/dev/video1`. For some IR cameras, starting the system already activates the (built-in) IR lights.

If you are unsure, either find out the name of the device file using a photo application like Cheese, or use the `v4l2-ctl` command from the *v4L-utils* package:

Listing 1: Edit Howdy Configuration

```
01 [core]
02 # Print that face detection is being attempted
03 # default: false
04 detection_notice = true
05
06 # The path of the device to capture frames from
07 # Should be set automatically by an installer
08 # if your distro has one
09 device_path = /dev/video0
10
11 [snapshots]
12 # Capture snapshots of failed login attempts
13 # and save them to disk with metadata
14 # Snapshots are saved to the "snapshots" folder
15 capture_failed = true
16
17 # Do the same as the option above but for
18 # successful attempts
19 capture_successful = true
```

```
$ v4l2-ctl --list-devices
```

Then copy the desired device, including the path to the configuration.

For face recognition using OpenCV, it does not matter whether the camera delivers normal monochrome or color images or IR images. For this reason, the standard laptop webcam is suitable for identification, unless you need genuine security.

The `snapshots` section in last part of the configuration file is also important. This section is where you specify whether Howdy saves an image at the time of authentication (Listing 1, lines 15 and 19). These snapshots help to analyze problems if face recognition did not work. Sometimes you can tell by looking at the image if there is insufficient light, if a reflection blinded the camera, or if parts of the face were obscured.

Howdy saves the image for each call as a JPEG file (Figure 1) in the `/usr/lib/security/howdy/snapshots/` directory; you will want to delete these images occasionally. If authentication fails, the software creates a joint image of all three attempts that it makes by default, labeled *FAILED LOGIN*. The smaller the value for *Best certainty value*, the better the recognition process worked. On the other hand, if the login is successful, Howdy only stores one image.

PAM Integration

To enable Howdy, you need to manually edit the PAM configuration files in the `/etc/pam.d/` directory. The data you store in these files determines where Howdy is used. Normally, this directory contains at least the files described in Table 3 – and often many more.

To always authenticate using Howdy first, add the line from Listing 2 at the beginning of the appropriate files. Listing 2 says that the Python program `pam.py` from the *howdy* package is used for logging in and that successful authentication with it is okay (keyword: `sufficient`). If the check fails, PAM starts the next module mentioned in the file.

Listing 2: Howdy Integration

```
auth sufficient pam_python.so /lib/security/howdy/pam.py
```

Table 3: PAM Files

File	Function
`sudo`	Controls access to the file of the same name
`system-login`	Active for each login
`system-local-login`	Active for local login
Prefix `user`	Active for user management
Prefix `group`	Active for user management
Prefix `ch`	Active when editing file permissions and owners

Setup and Training

After the installation, the Howdy configuration involves two steps. After the first step, the system recognizes faces in the input data, but without differentiating them. Differentiating between people requires separate training in the second step.

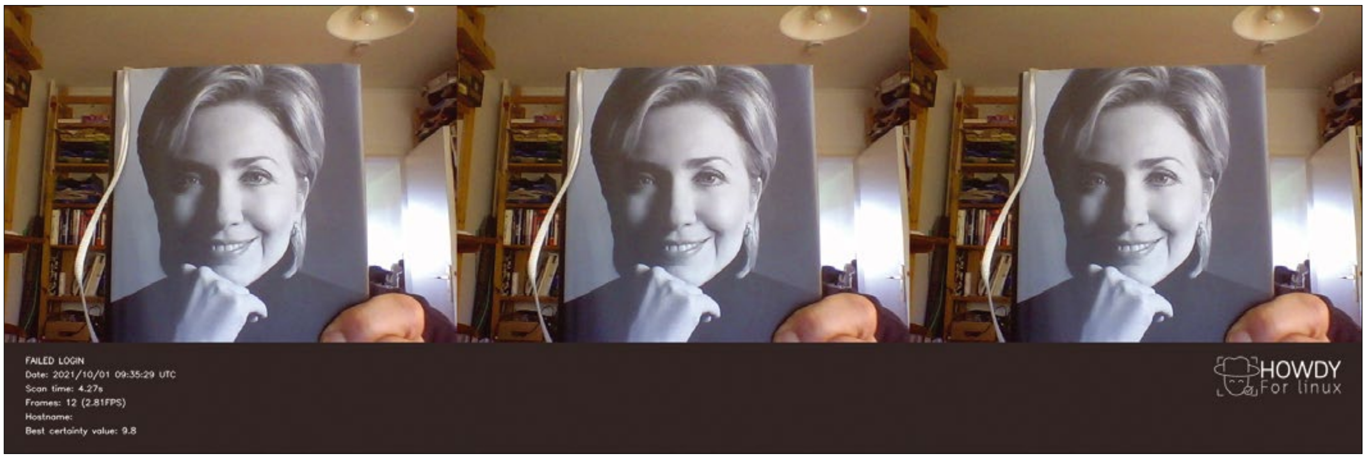
In this training, the neural network used by OpenCV learns specific features that represent the user's face. Since Howdy does not just learn static images from the video stream, but short sequences, it is usually not a problem if you blink briefly or move your head a bit. In fact, such slight movements seem to improve recognition rates in real-world use. It takes several repetitions to anchor a solid base in the neural network.

After configuring the hardware and before training, a short test is recommended. Use the command from the first line of Listing 3 to start a special mode that displays the video stream in a new window. Initially, the goal is not to identify a specific face, but to recognize faces in a generic way. If successful, OpenCV marks the area with a red circle (Figure 2). In this way, you can test whether the lighting conditions are sufficient, the position fits the camera, and no disturbing reflections occur. Pressing the Enter key closes the window again and exits the program.

Alternatively, you can try the `snapshot` subcommand, which Howdy uses to capture the images that you analyze afterwards (Listing 3, second line). This option is useful if the program does not recognize faces in test mode. Howdy saves the test images, which are always black and white, in the current directory.

Listing 3: Test Mode

```
$ sudo howdy test
$ sudo howdy snapshot
```



```
FAILED LOGIN
Date: 2021/10/01 09:35:29 UTC
Scan time: 4.27s
Frames: 12 (2.81FPS)
Hostname:
Best certainty value: 9.8
```

Figure 1: When you log in via Howdy, the program saves an image in the system. In the test, the bright light reflex on the right edge probably confused the software.

Once everything is working, you’re ready for the actual deployment. First capture the features of the faces of the users who are allowed to use the technology (Listing 4). Howdy requires root privileges for this task. The program is installed as `/usr/bin/howdy`, a symbolic link to `/lib/security/howdy/cli.py`, so it reads and writes the data to the `/lib/security/howdy/` directory. To keep the rate of failure low, choose a setting for the recording that is as much as possible like the place you will use it.

Adding new profiles has a downside: The more profiles you feed into the neural network, the slower it reacts. This makes it important to find a good compromise between recognition rate and speed. It is best to load only a few profiles at first, maybe two or three. If errors occur during authentication, add another profile each time. In the test, the login worked in most cases with upward of 10 profiles, as long as the conditions between training and application did not differ too greatly – and this means both the exposure situation and the angle to the camera.

The `list` option tells Howdy to display the stored profiles (Listing 5, line 1). This list does not offer an evaluation of the profiles. It is not possible to see how well the software recognized a face or how big the match is for a profile. You specify the `Label` when saving the profile. If you don’t specify it, the program creates these entries on its own, numbering them consecutively and prefixing each with the string `Model #`. Howdy also assigns the `ID` in the first column automatically. The ID allows Howdy to uniquely identify individual profiles, for example, in order to delete them (Listing 5, line 17).

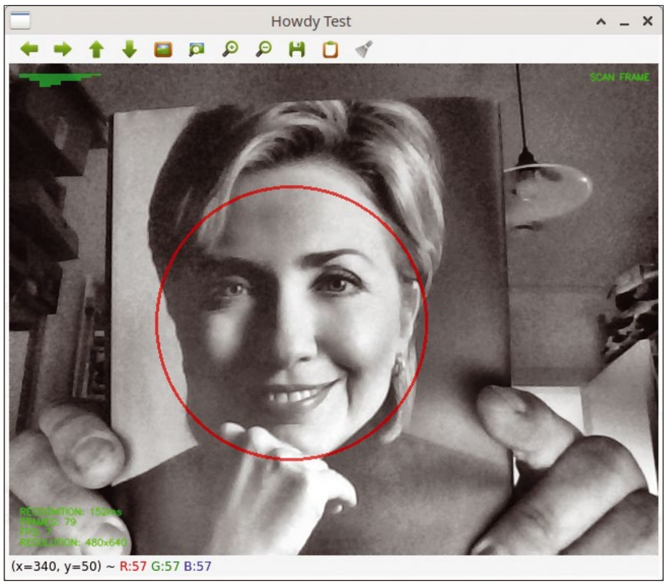


Figure 2: OpenCV detects a face in a well-printed image.

Listing 4: Storing Facial Features

```
$ sudo howdy add
NOTICE: Each additional model slows down the face
        recognition engine slightly
Press Ctrl+C to cancel
Adding face model for the user Bob
Enter a label for this new model [Model #12]
    (max 24 characters):
Bob
Please look straight into the camera
Scan complete
```

Listing 5: Listing Profiles

```
01 $ sudo howdy list
02 Known face models for User:
03   ID  Date                 Label
04   0   2021-09-22 11:23:44  User
05   1   2021-09-22 11:23:55  User
06   2   2021-09-22 11:24:29  User
07   3   2021-09-22 11:26:47  User
08   4   2021-09-22 19:20:36  User
09   5   2021-09-22 19:20:48  User
10   6   2021-09-22 19:21:01  User
11   7   2021-09-27 09:57:20  User
12   8   2021-09-27 09:57:39  User
13   9   2021-10-12 09:17:44  Model #10
14   10  2021-10-12 09:19:08  Model #11
15   11  2021-10-12 11:59:37  Model #12
16   12  2021-10-12 13:22:28  User
17 $ sudo howdy remove 11
```

Disabling Warnings

OpenCV and GStreamer cooperate without any problems for the most part. However, OpenCV proves to be quite talkative in the default installation and starts outputting warnings in the shell (Listing 6). This is annoying and unnecessary, and it makes working with the software more difficult. One simple remedy is to prevent warnings from being output. To disable warnings, set an appropriate environment variable in the shell in which you start Howdy (`OPENCV_LOG_LEVEL=ERROR`).

Security

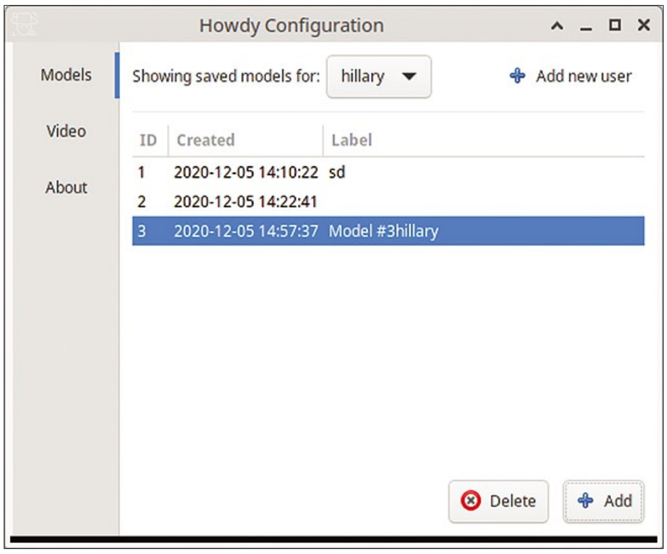
As with all biometric applications, it is possible to fool Howdy. Figure 1 shows that a well-made image can pass as a face even if the size does not really fit. In the example, however, several attempts were required, and we had to permanently move the book cover used as a fake slightly in order to successfully record a profile. OpenCV’s network is particularly sensitive to nodding.

On the Howdy homepage, developer Lem Severein points out the danger of manipulation and emphasizes the need to use a good IR video camera. But even a good IR camera could be fooled with some additional effort, just as fingerprint sensors can be fooled. The German Chaos Computer Club (CCC) has demonstrated problems with biometric authentication several times.

Severein came up with a useful extension to improve security known as rubber stamps [5]. Whenever Howdy recognizes a face – and only then – you can call predefined additional routines. These routines can then add additional tests, such as whether the user has pressed a certain key. If the additional condition is met, the software evaluates the authentication as correct. The mechanism is still missing in the current stable Howdy v2.6.1, but it is already available in the Git repository.

A separate tutorial shows how to implement appropriate rules (stamp rules). Howdy uses the possibilities of the OpenCV library. You can, for example, use a nod or a shake of the head as confirmation or negation. You are even allowed to define a minimum number of nods to confirm the authentication.

Severein has been working on Howdy 3.0 for more than a year, but a release date has not been set yet. In addition to the rubber stamps, version 3.0 will probably include a graphical user interface (GUI). A preview is available in the form of the developer version (Figure 3). To start the interface, you need the Python `elevate` module from the *python-elevate* package. Then change to the `src/` directory and call `./.init.py`.



ID	Created	Label
1	2020-12-05 14:10:22	sd
2	2020-12-05 14:22:41	
3	2020-12-05 14:57:37	Model #3hillary

Figure 3: A preview of Howdy 3.0, which will probably include a GUI.

Conclusion

Howdy shows where the world is headed: Self-learning systems and biometric data for authentication will probably play an important role in the future. So far, however, the special hardware requirements mean that Howdy is more of a proof of concept than an actual authentication solution suitable for industrial use. Having said this, you can use rubber stamps to increase security in practice. ■■■

Info

[1] Howdy: *https://github.com/boltgolt/howdy*

[2] OpenCV: *https://opencv.org*

[3] Switching on IR lamps: *https://github.com/EmixamPP/linux-enable-ir-emitter/wiki/Semi-automatic-configuration*

[4] Howdy in the AUR: *https://wiki.archlinux.org/title/Howdy*

[5] Rubber stamps: *https://github.com/boltgolt/howdy/wiki/Rubber-Stamp-Guide*

Listing 6: OpenCV Warnings

```
[ WARN:0] global /build/opencv/src/opencv-4.5.3/modules/videoio/src/cap_gstreamer.cpp (2056) handleMessage OpenCV |
      GStreamer warning: Embedded video playback halted; module source reported: Could not read from resource.
[ WARN:0] global /build/opencv/src/opencv-4.5.3/modules/videoio/src/cap_gstreamer.cpp (1034) open OpenCV |
      GStreamer warning: unable to start pipeline
[ WARN:0] global /build/opencv/src/opencv-4.5.3/modules/videoio/src/cap_gstreamer.cpp (597) isPipelinePlaying OpenCV |
      GStreamer warning: GStreamer: pipeline have not been created
```

A cohesive open source ecosystem

Haiku

Haiku, an outgrowth of the proprietary BeOS, offers a cohesive open source operating system as an alternative to Linux. *By Bruce Byfield*

Around the turn of the millenium, the proprietary BeOS included many features that made it stand out from other operating systems. Despite a determined effort, it never gained a commercial foothold. However, it did capture some enthusiastic fans. After production ceased in 2002, a half dozen attempts to continue or clone BeOS sprang up, of which only Haiku exists today (Figure 1).

Although officially in its third beta, in 2022, Haiku is a stable operating system with a somewhat archaic look to its icons and widgets. However, with a little help from Plasma applications and standard software like LibreOffice, it is generally usable for modern computing purposes. Its speed is compelling, and even some minor features, like the ability to tab multiple applications in a single window or the use of a drop-down list to display open windows, make clear that Haiku is more than an exercise in nostalgia. To learn more about Haiku's modern relevance, I contacted one of the few people sometimes paid to develop Haiku, a developer and project mentor who prefers to go by the handle waddlesplash (or sometimes Mr. waddlesplash). As you might expect, some of his opinions might be surprising for many Linux users.

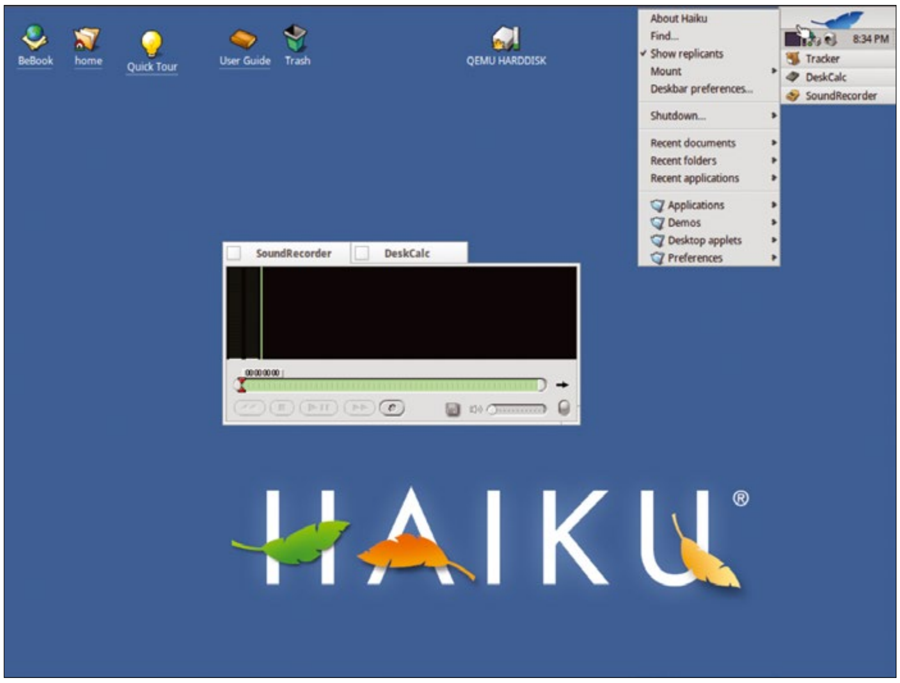


Figure 1: The default Haiku desktop: Note the two apps shown as tabs in the same window and listed in the Tracker on the upper right.

Linux Magazine (LM): What was the original appeal of BeOS?

waddlesplash (w): In the 1990s, BeOS was particularly advanced for its time. Not only did it natively utilize multiprocessor computing years before it became ubiquitous on the desktop, but its native APIs and toolkits were C++, and they were natively multithreaded for highly responsive user interfaces. They also had very advanced multimedia for the time. Moreover, the robustness of BeOS was almost legendary.

LM: I understand that for a long time Haiku maintained backwards compatibility with BeOS. Did backwards compatibility help or hamper development?

w: A bit of both. Having backwards compatibility with BeOS made it so that initially we had a fixed target to aim for and a baseline to compare things to. Nowadays it is a bit more of a hindrance, but we have dropped it in a lot of areas where it mattered less. For example, the kernel no longer supports loading BeOS kernel drivers, and we compile a lot of applications with the newer, non-BeOS-ABI-compatible GCC even on the "BeOS-compatible" builds.

LM: Do any of the original advantages of BeOS remain?

w: By and large, the rest of the world has caught up with the major innovations BeOS had. Haiku still has some innovations of our own, though. The biggest reason that Haiku is still something a lot of us believe in and still work on is because it is the only serious contender for a *cohesive* open source operating system for personal computing.

LM: What do you mean by a cohesive operating system?

w: The general argument we make on Haiku is that such problems Linux has are inherent to the ecosystem the Linux desktop has chosen for itself. There is not one "Linux desktop" project (or even a few competing ones); there's X11 (or now Wayland) for the display server, various differing options for a window manager, [and] Gnome or KDE or Xfce or whatever else for the desktop environment (and these are themselves huge projects with often competing interests internally). That's before we even get to ALSA, PulseAudio, systemd, or the multiple dozen other components you have to mix together in order to come up with something that actually can boot and display a web browser.

[As an example of the complications Linux's setup can cause,] take the bug reporting process on Linux. You have to begin by first establishing what project has responsibility for the application in question with the bug (is it Gnome? is it one of Gnome's sub-projects? is it really in PulseAudio, and Gnome just is displaying the error message?) which requires you to basically be an expert in the first place. Then once you have reported it, you may find out that the bug has already been fixed in a newer version your distribution does not have or is due to an incompatibility created by your distribution's packagers, or may well be a legitimate bug but is unreplicable outside of your distribution, so the developers do not care to bother with it. [By contrast,] Haiku is a fully cohesive ecosystem. There are of course still some issues with sorting out when a problem in an application is due to a bug in Haiku, a bug in our port of the application, or the application itself, but these are much easier to resolve when the boundaries between projects are much clearer. Bugs or feature requests for the base system itself are even easier: Whether it's really a bug in the UI toolkit, the graphics server, or the kernel, it's our responsibility, and we just need to agree amongst ourselves where [is] the best place to solve it. This means there's a lot less wasted effort and a lot more efficient development. If someone spots a minor problem in the file manager, a developer can post a patch for it, have installable test builds available within hours, and once it's merged, it will be in the binary update repositories for the "nightly" channel the next day. And everyone who updates their OS will get it. Whereas on Linux, if you report some issue with a default application and it gets fixed, it could take a year to trickle down into your local distribution, and you would otherwise have to resort to compiling it yourself or using an AppImage.

LM: Haiku's latest release is officially the third beta. Will Haiku ever have a general release?

w: "Beta" means we have replicated all functionality of BeOS R5 as well as implemented all of the features we had on our to-do list for R1. R1 will be ready "when it's ready." At this point, Haiku's desktop environment is probably on average more stable than, say, KDE 4.0 was at the time of its initial release, and kernel stability and hardware is not too shabby either.

LM: What sort of user is Haiku likely to appeal to? What features would you point out to new users?

w: [For] anyone who already runs desktop Linux or BSD and enjoys experimenting with beta-quality software, and Haiku sounds intriguing to you, I would encourage you to try it out – especially if you have some slightly older hardware that Linux seems a bit sluggish on these days. Haiku is still pretty resource-light and can often make the most out of slightly older hardware. (Newer hardware is fine, too, though: I run Haiku on bare-metal on a 16-thread Ryzen box!)

One of the more technically interesting features that I would mention is our package manager. Unlike Linux packages, which are just archives extracted at installation time, Haiku packages are block-compressed filesystem images. On boot, the bootloader loads the kernel from the main system package, and it then mounts all your installed packages during the boot sequence. Installing packages just means they get copied into the system packages directory and then mounted, or on de-installation [the package] is copied into an "old state" folder. So if you take a bad update or install a package which winds up breaking your system, you can ask the bootloader to start you into one of those older states. Or you can boot into a different installation and replace the set of installed packages altogether. Besides making package installation much faster, as well as making certain failure conditions impossible (no more "installed files database" to lose or corrupt like on Linux), or at least easier to resolve, this deep integration with the system is pretty typical of Haiku.

LM: What sort of kernel does Haiku use?

w: The Haiku kernel is a modular monolithic kernel. The kernel itself handles the basics of system calls, I/O, and memory management, and basically everything else (including the PCI bus, the filesystem drivers, the network stack, the USB stack, etc.) [is] all dynamically loaded modules.

We have been shifting some things into userland, though. Newer USB device drivers have been pushed to userland, and the upcoming work on 3D acceleration puts almost all of the GPU management logic in userland, with only a small amount of code in the kernel-side driver. There also used to be a way to run the network stack in userland, too, but that has bitrotted by now. I would say that most of the developers' sensibilities lean more towards the microkernel side of things than not, these days, but it would be a lot of work for not much benefit to do that refactor at this point. Maybe someday …

LM: Are any non-free components used in Haiku?

w: We used to ship a number of closed-source BeOS applications in the default install (for which we had permission from the original developers to do so), but these have all been open sourced, with maybe one minor library as an exception, at this point. I think the only other non-free components are the WiFi firmware blobs, but you can remove them if you have a WiFi chip that does not need them (or you don't need WiFi.)

Haiku itself is mostly under the MIT license, though, and overall we are not at all averse to commercial usage. There are (or were) at least one or two companies that used to run BeOS that were deploying Haiku commercially.

LM: How is Haiku organized and governed?

w: The project has voting team members, generally those with commit rights, who determine the direction of the project. If there are disagreements, theoretically we would put it to a vote. But in practice we are always able to come to some kind of consensus and proceed without an official vote. I can't recall any formal votes besides ones to grant commit rights while I have been a voting member these past six years.

There is also Haiku, Inc., a 501(c)3 non-profit which manages everything related to finances, but the board of the Inc. does not direct the project (though most of the board are voting members of the project).

I would say we have about a dozen or two regular contributors of code and ports at present, and a few hundred regulars on the forums and IRC who use the system on some kind of regular basis. Honestly, the fact that Haiku is able to at the very least not lose ground compared to desktop Linux's feature set and has been slowly gaining it the past few years is a testament to how much we have done with very little. Before a few months ago when I was brought on as a contractor to work on Haiku full time, I would say we had maybe one to two person-months of work done per month across the whole ecosystem, at best; it was probably less. If we had even a tiny fraction of the amount of money and time poured into the desktop Linux ecosystem each year, I would have to imagine we would catch up completely and then go beyond it very rapidly.

LM: Where is Haiku headed in the future?

w: Who knows what the future holds? Whatever is needed for personal computing, or at least whatever we and our users need, we'll try to support, as we have development time for. I realize that's a vague answer, but that's because our plans are still vague. Haiku has survived for over two decades now plodding along and doing whatever is next, and I imagine we will continue doing so for as long as we can.

LM: Would you like to say anything else?

w: I started volunteering for Haiku in early high school, and I learned a massive amount in working on it all these years from the other developers and contributors – not just in technical terms, but also in community organization, how to mentor new developers, and too many other things to count. I'm very grateful for all that I've learned, and the patience with which it was taught, and now I get paid to pay it forward by working on Haiku and mentoring the current newcomers! Overall, it is just a great project to be a part of. ■

DrupalCon

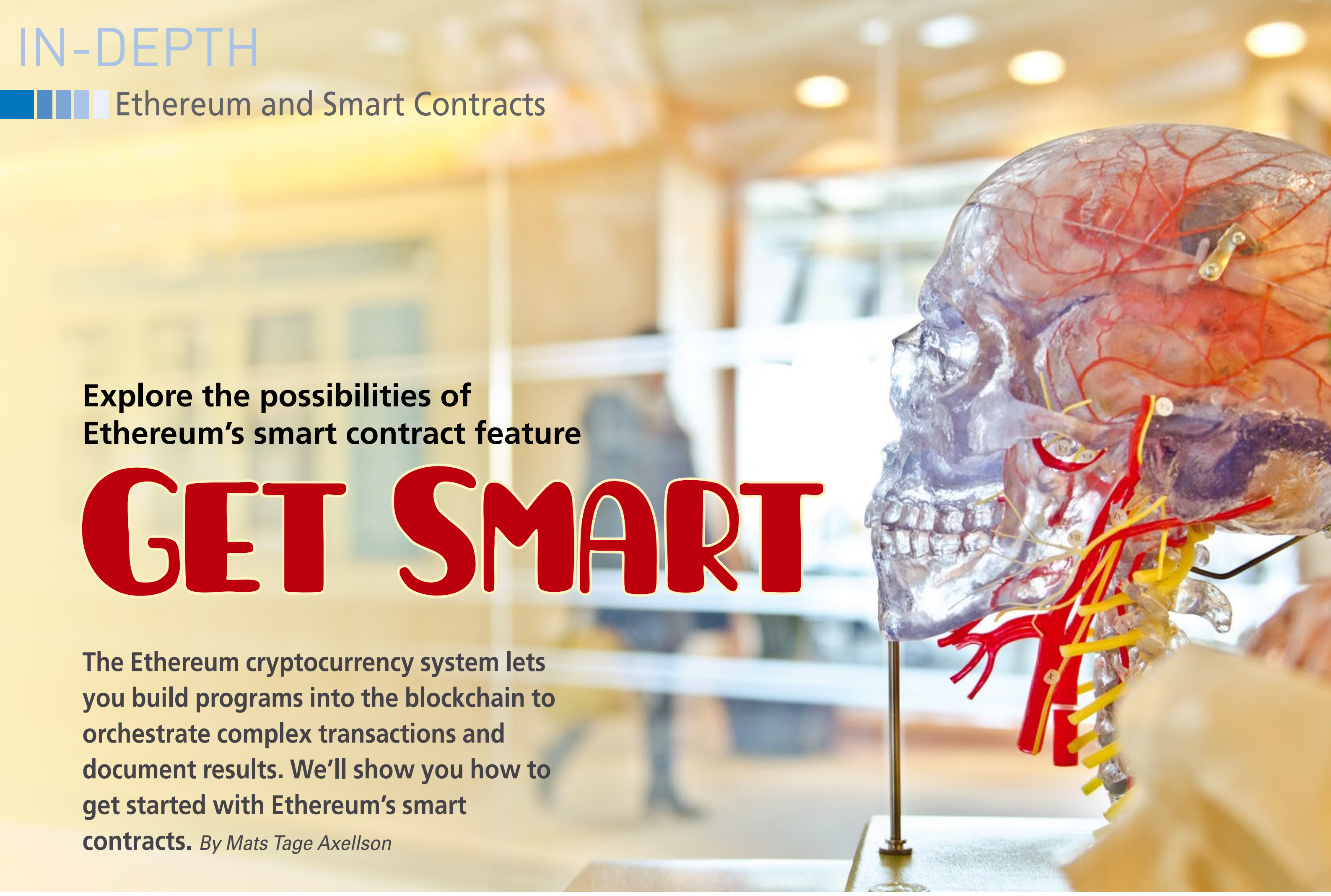
PORTLAND**2022**

25-28 APRIL

Oregon Convention Center

events.drupal.org/portland2022

Join us for DrupalCon Portland, where the people who make amazing digital experiences possible come together to make them even better.



Explore the possibilities of Ethereum's smart contract feature

Get Smart

The Ethereum cryptocurrency system lets you build programs into the blockchain to orchestrate complex transactions and document results. We'll show you how to get started with Ethereum's smart contracts. *By Mats Tage Axellson*

Digital currencies have had quite a lot of buzz around them in recent years, with Bitcoin being the most famous. Satoshi Nakamoto, the mysterious creator of Bitcoin, had the idea to use Bitcoin as a store of value, like gold. For that reason, Bitcoin was designed for the limited purpose of trading and holding value.

When Vitalik Buterin and others designed the first version of Ethereum [1], they wanted to extend the blockchain concept to include other kinds of transactions. Ethereum is built around the concept of a *smart contract*. A smart contract is a program embedded in the blockchain that automatically manages, controls, and documents actions taken for a predetermined purpose. A smart contract allows the creator to implement a contract or agreement without the need for traditional oversight and enforcement.

Bitcoin is thought to have some rudimentary support for smart contracts in the context of its role as a digital currency – to support features such as escrows and multisignature accounts, but Ethereum expands the concept to support a much wider range of transactions and arrangement. For instance, Ethereum was the first blockchain technology to support non-fungible tokens (NFTs) [2], and Ethereum is still the most popular tool for creating NFTs.

This capacity for smart contracts makes Ethereum an ideal candidate for supporting decentralized autonomous organizations (DAOs), blockchain-based insurance, and other systems that require verification of real life activities.

The Ethereum Blockchain

The principle of a blockchain is the same for all cryptocurrencies. If you have spent more than a few hours with programming, you will know about chained lists. The idea is that the first item points to the next, creating a list that grows according to your needs.

With a blockchain, everyone and anyone can download the chain and add to it. How do you make sure the newest block is correct? How do you know that people you do not know haven't changed the earlier blocks? The blockchain has to be tamper proof.

To handle this, every block after the first points backwards with a cryptographic hash. That hash will depend on the contents of the earlier block. Not only that, all blocks' hash values depend on the earlier block all the way back to the first block. That first block is called the genesis block, and the complete system is what is called a Merkle Patricia tree.

Wait, is it a tree or a chain? Actually it is a little of both. If the blockchain were just a chain, any verification would have to go through the entire chain. The entire Ethereum Virtual Machine (EVM) would grind to a halt, and no one could determine the state of the system.

In essence, the whole Ethereum system is one state machine. A state machine is a construct that changes its state depending on input. In this case, the input is every transaction made in the EVM. It sounds odd but the state is not a single value as it is for simple applications. Instead, the state has many values that are constantly changing. The state is expressed in tree form.

Ethereum has three trees that you can follow to verify any actions on the chain. These trees can verify ownership and also verify that the tree itself is correct. The trees are `state`, `transactions`, and `receipts`. Thanks to the trees being separate, it is much easier to use a small part of the chain to verify that everything is correct.

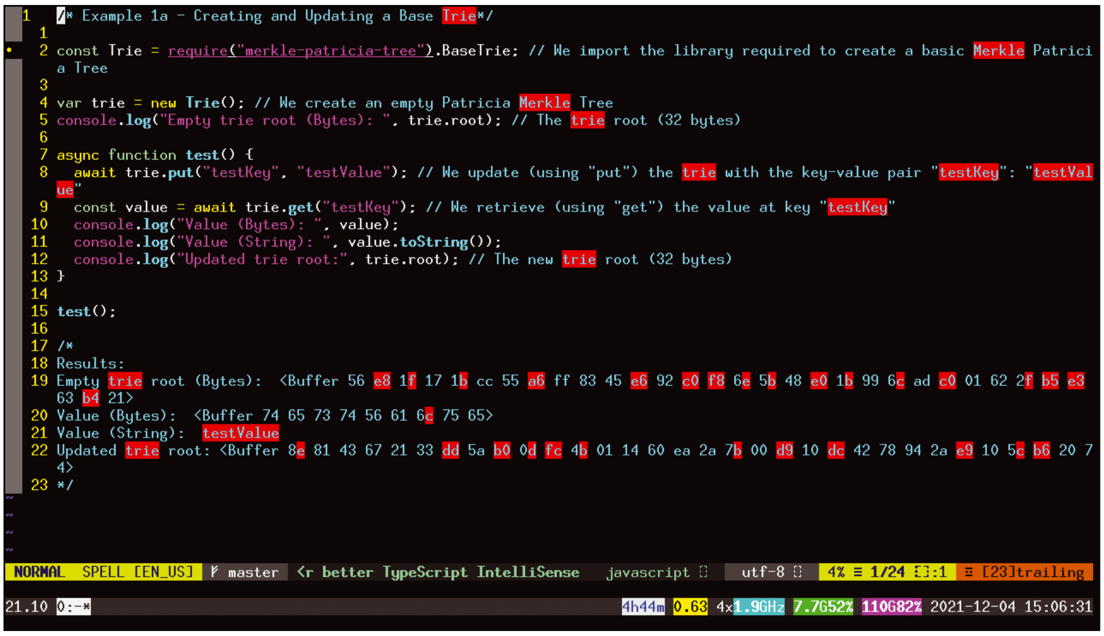
Each tree has a root, `stateRoot`, `transactionsRoot`, and `receiptsRoot`. Their values are in the header of each block of the Ethereum blockchain. This approach enables clients to verify all three without querying the entire tree. A pruned tree is around 1.2TB, and a full tree is around 6TB. As you can see, it is not easy to have a full node.

When you want to use your ether (ETH – the cryptocurrency used with Ethereum), or verify anything about the Ethereum blockchain, you need a client. A full client with a full node will require too much disk space for most systems. A *light node* requires only a small fraction of the tree and still allows you to make advanced queries on the tree. You can use a light node to balance your own account, check if another account exists, and test run a contract. Testing is important to ensure that you don't fall victim to a fraudulent contract.

The trees have different functions, and you need them for different tasks. Your client needs to use the `state` tree for checking account-related things and testing contracts.

The `transactions` tree will be updated often but never edited. In contrast, the `state` tree changes frequently, because it contains all accounts and their balances.

The `receipts` tree only keeps a record of the effects of transactions, so it is also updated but not edited. If you want a more hands-on approach to handling Merkle-Patricia trees, check out the tutorial online [3]. You can download the code from the tutorial and play around with it yourself. The first example is in Figure 1, which shows how to initiate a tree.



```
/* Example 1a - Creating and Updating a Base Trie*/

const Trie = require("merkle-patricia-tree").BaseTrie; // We import the library required to create a basic Merkle Patrici
a Tree

var trie = new Trie(); // We create an empty Patricia Merkle Tree
console.log("Empty trie root (Bytes): ", trie.root); // The trie root (32 bytes)

async function test() {
  await trie.put("testKey", "testValue"); // We update (using "put") the trie with the key-value pair "testKey": "testVal
ue"
  const value = await trie.get("testKey"); // We retrieve (using "get") the value at key "testKey"
  console.log("Value (Bytes): ", value);
  console.log("Value (String): ", value.toString());
  console.log("Updated trie root:", trie.root); // The new trie root (32 bytes)
}

test();

/*
Results:
Empty trie root (Bytes):  <Buffer 56 e8 1f 17 1b cc 55 a6 ff 83 45 e6 92 c0 f8 6e 5b 48 e0 1b 99 6c ad c0 01 62 2f b5 e3
63 b4 21>
Value (Bytes):  <Buffer 74 65 73 74 56 61 6c 75 65>
Value (String):  testValue
Updated trie root: <Buffer 8e 81 43 67 21 33 dd 5a b0 0d fc 4b 01 14 60 ea 2a 7b 00 d9 10 dc 42 78 94 2a e9 10 5c b6 20 7
4>
*/
~
~
~
~
NORMAL  SPELL [EN_US]  master  <r better TypeScript IntelliSense   javascript   utf-8   4% ≡ 1/24 :1  ≡ [23]trailing
21.10 0:-*                                                    4h44m 0.63 4x1.9GHz 7.7G52% 110G82% 2021-12-04 15:06:31
```

Figure 1: Code example from the Merkle Patricia tutorial collection.

Transactions

The simplest transaction is to pay someone an amount. You can try this by making a payment between your own accounts – or better yet on a testnet. Each transaction will cost you some amount of gwei (a gwei is the smallest unit of Ethereum currency, equivalent to one billionth of an ether.) The actual cost for a transition will depend on the *gas price*. According to Wikipedia, "When creating a transaction, the sender must specify a *gas limit* and *gas price*. The *gas limit* is the maximum amount of gas the sender is willing to use in the transaction, and the *gas price* is the amount of ETH the sender wishes to pay to the miner per unit of gas used."

Ordinary wallets set the gas price automatically in a unit of gwei. When you do more advanced transactions, you can increase the gas price to give miners more incentive to do your job first.

Most people use browser extensions for transactions. MetaMask is a great Ethereum client, and it is very common. In the MetaMask wallet, you can send and sometimes choose the gas price yourself. MetaMask also gives you the chance to speed up the transaction while it is pending. You can also ask for funds by giving out your public address.

For Linux desktop, you have Jaxx, Exodus (Figure 2), and MyCrypto wallets. These wallets have more functions and a fancier interface, which includes graphs for exchanges. More importantly, you can have wallets from different blockchains,

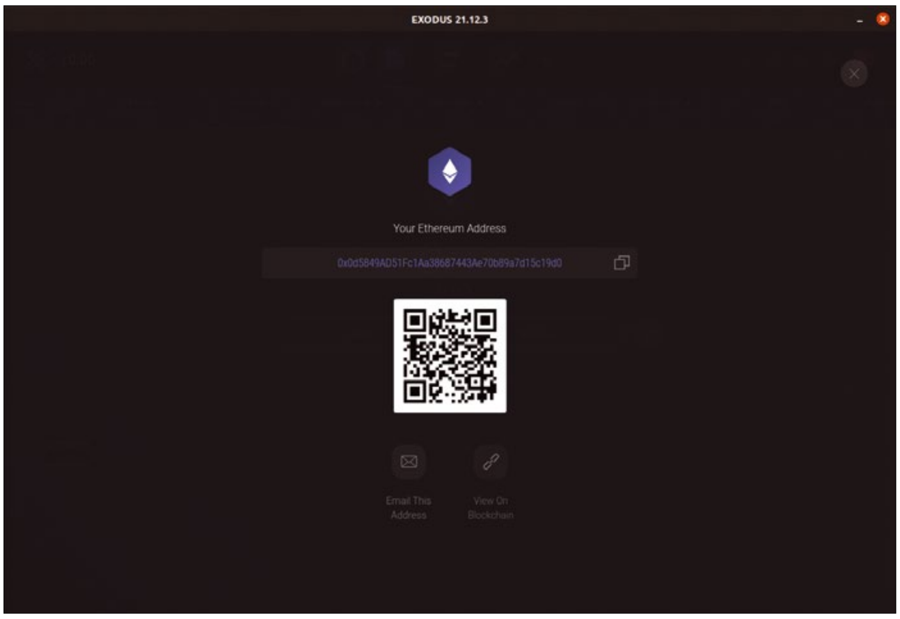
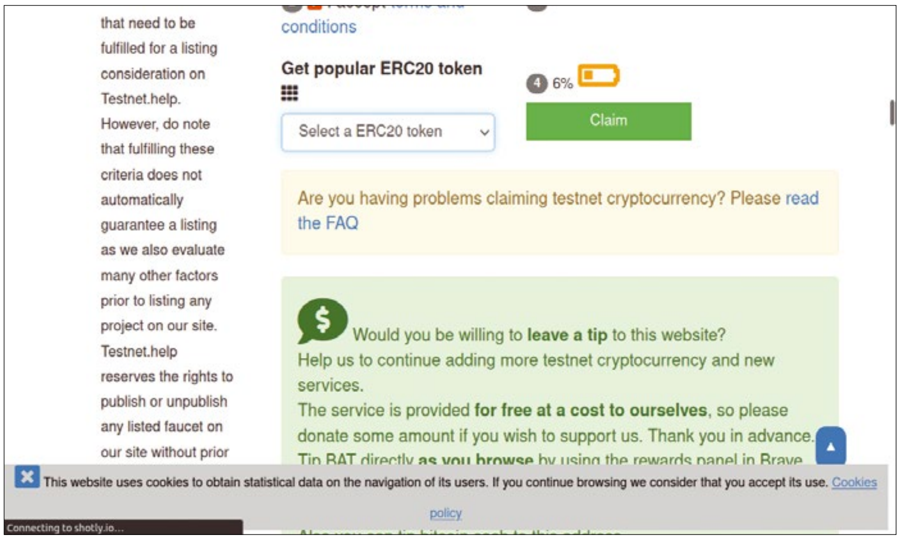


Figure 2: When using Exodus on the Linux desktop, you get a QR code for people to send you funds.



that need to be fulfilled for a listing consideration on Testnet.help. However, do note that fulfilling these criteria does not automatically guarantee a listing as we also evaluate many other factors prior to listing any project on our site. Testnet.help reserves the rights to publish or unpublish any listed faucet on our site without prior

conditions

Get popular ERC20 token

Select a ERC20 token

4 6%

Claim

Are you having problems claiming testnet cryptocurrency? Please read the FAQ

Would you be willing to **leave a tip** to this website?
Help us to continue adding more testnet cryptocurrency and new services.
The service is provided **for free at a cost to ourselves**, so please donate some amount if you wish to support us. Thank you in advance.
Tip BAT directly **as you browse** by using the rewards panel in Brave

This website uses cookies to obtain statistical data on the navigation of its users. If you continue browsing we consider that you accept its use. Cookies policy

Connecting to shotly.io...

Figure 3: When you ask for tokens on the testnet, try to do it on Wednesdays. The low power icon means there is too much traffic to give more.

not just Ethereum. Note also that transactions happen when a contract is created and during execution of contracts.

Smart Contracts

According to the Ethereum website, "A 'smart contract' is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain."

Smart contracts can solve many problems. They can do such diverse things as creating new currencies and facilitating crowdfunding. The Basic Attention Token (BAT) system used by digital advertisers for tracking attention time is an example of an Ethereum-based blockchain technology that leverages the power of smart contracts.

The Ethereum documentation goes on to explain "Smart contracts are a type of Ethereum account. This means they have a balance and they can send transactions over the network. However they're not controlled by a user; instead they are deployed to the network and run as programmed. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible."

In Ethereum, smart contracts are special accounts that must first be created (minted) by user accounts. The creator of a contract must have written the code and also must have an account and currency (ether). When the user creates a contract, the code can set values that are immutable once it is on the blockchain.

Once the contract is on the chain, any other users can call it at a cost. The cost will vary depending on the contract. A smart contract could also be an auction for an NFT.

You can imagine how a program would execute an auction. Bids would be received and accepted. Then at some predefined decision point, the program would select the highest bidder as the winner. At that point, all losing bids would revert back to the bidder. After that, the program would transfer the funds to the initiating user. This type of auction can be embedded directly into the blockchain using a smart contract.

A developer needs to write the code for the contract, test it, and then deploy it to the blockchain. The Ethereum community has specialized development tools for coding and deploying smart contracts. Coders can also access a testnet that acts like the main Ethereum blockchain except that it has no value. (Tokens for the testnet are available online [4]). You can also use the testnet to get used to regular transactions (Figure 3).

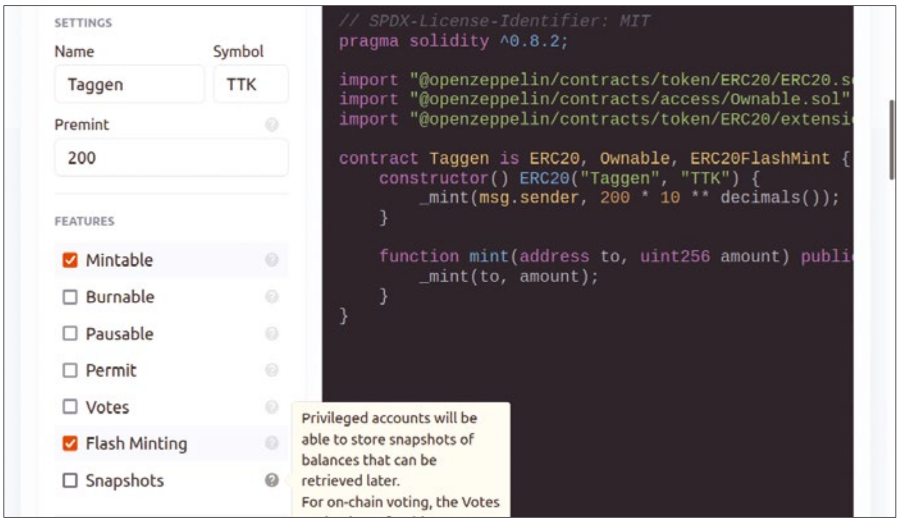
Solidity, the Language!

You can program smart contracts in many languages, but the most common is Solidity [5]. Solidity looks and behaves much like the C language in that you have constructors and functions, and you need to set a variable's type.

Every instance is a contract, and each contract can have one and only one constructor. Functions and values get a visibility setting, telling the compiler which other functions can use them. The `public` setting, for example, makes a function available to any function on the entire blockchain. As you can see in Figure 4, Solidity also lets you import frameworks. There are other languages under development, but Solidity is a good place to start if you want to learn about programming smart contracts.

Development Environments

To truly start developing, you need an IDE; Cakeshop is a good choice. The Cakeshop IDE acts as a daemon, running a local chain that you can do pretty much



SETTINGS

Name	Symbol
Taggen	TTK

Premint: 200

FEATURES

[✓ Mintable]
☐ Burnable
☐ Pausable
☐ Permit
☐ Votes
[✓ Flash Minting]
☐ Snapshots

Privileged accounts will be able to store snapshots of balances that can be retrieved later.
For on-chain voting, the Votes

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.2;

import "@openzeppelin/contracts/token/ERC20/ERC20.s
import "@openzeppelin/contracts/access/Ownable.sol"
import "@openzeppelin/contracts/token/ERC20/extensi

contract Taggen is ERC20, Ownable, ERC20FlashMint {
    constructor() ERC20("Taggen", "TTK") {
        _mint(msg.sender, 200 * 10 ** decimals());
    }

    function mint(address to, uint256 amount) publi
        _mint(to, amount);
    }
}
```

Figure 4: On the right, you can see the Solidity source code for importing Zeppelin, a framework for standards.

anything with. (See the box entitled “Setting Up Cakeshop.”) You also have the option of putting many nodes up on your local network. In the interface, you will find the basic stuff you need for interacting with a blockchain. You can create accounts and send funds back and forth on the test chain. Follow up by looking at the chain explorer and see how the transaction worked. There is also a sandbox, where you can deploy your own contracts and test all functions.

A smoother way to get started might be to use the Remix IDE online service [8]. Remix lets you load files from your local disk or even directly from GitHub. The features are similar to Cakeshop, but Remix does support plugins. The basic plugin is the Solidity plugin, which gives you full support for the language.

Remix offers many other plugins to choose from, including tutorials, verification tools, and example code. The code covers the simplest storage all the way up to distributed applications. For the really advanced, you also have services to reach outside information (called *oracles*).

Figure 7 shows the Remix IDE. On the left, you can choose the file explorer, compiler, and whatever plugins you wish to include. To the right, you have the code and a console.

A Simple Contract

Some basic contracts show up in the default view of Remix. See Listing 1 for an example that shows how the basic syntax works.

This code only contains simple function calls and no constructor, but it is an excellent start for understanding what you need to get right. The top line is not necessary, but many compilers will complain without it. The second line limits which compiler you can use.

Listing 1 has one contract function called `Storage`; all your code goes inside this function.

Setting Up Cakeshop

The developers chose to use Java for Cakeshop. The original place to pick it up is GitHub [6], where you can find a `.war` file and the source code. Using this binary, you can start it easily enough, but you need to define the nodes. If you just want to see how Cakeshop works, use the snap; it is still in version 0.10.0, so go with a newer version when you have figured out the configuration of the nodes. The snap version has defined nodes built-in, so it starts with a single command.

```
$ sudo snap install cakeshop
```

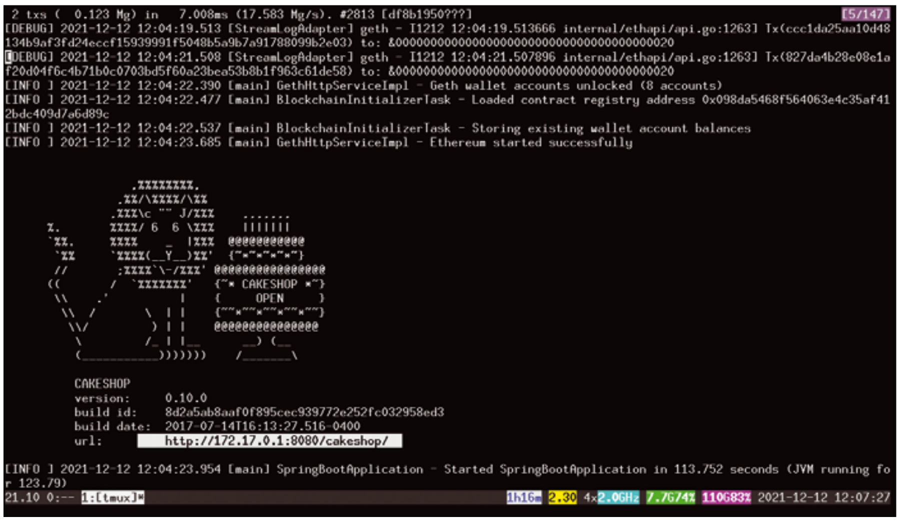
Wait for the install to finish, and run the following command on the command line:

```
$ cakeshop
```

With this setup, you can see what address you need to look at to use the web interface. Usually it would be *http://172.17.0.1:8080/cakeshop/*, but you must look in terminal output to be certain (Figure 5).

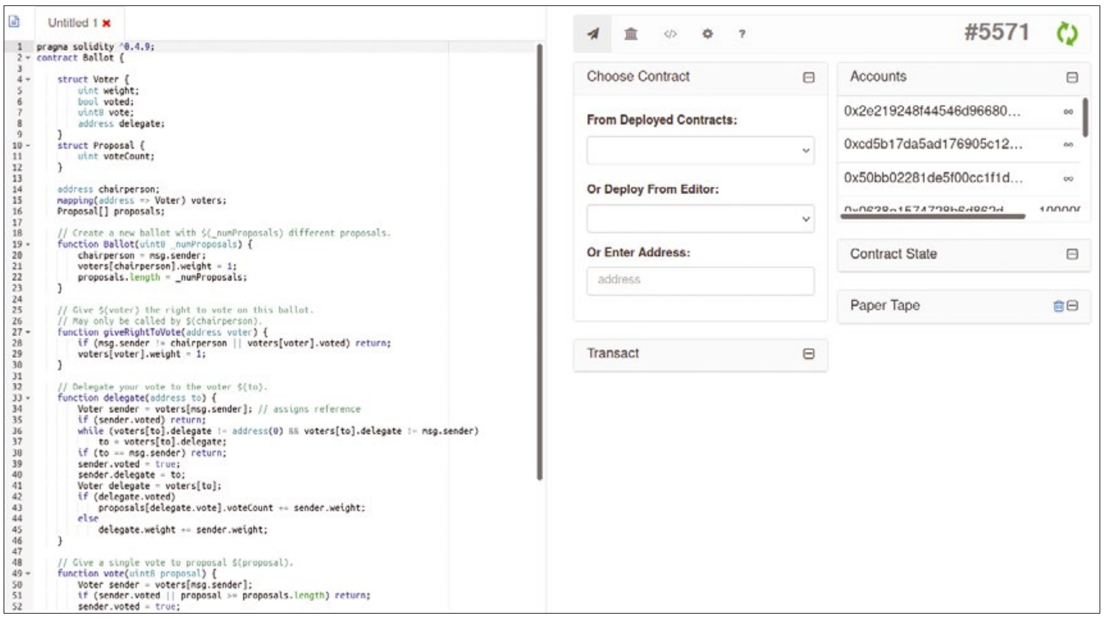
You can find more install options on the Cakeshop home page [7]. The page describes several ways to start a Docker instance.

When you have it all installed, you can see all components of a blockchain and even interact with the chain. This blockchain is your own, not the public one! Use the block explorer to investigate what your actions do. You will also have unlimited currency. You can see in Figure 6 what the sandbox looks like.



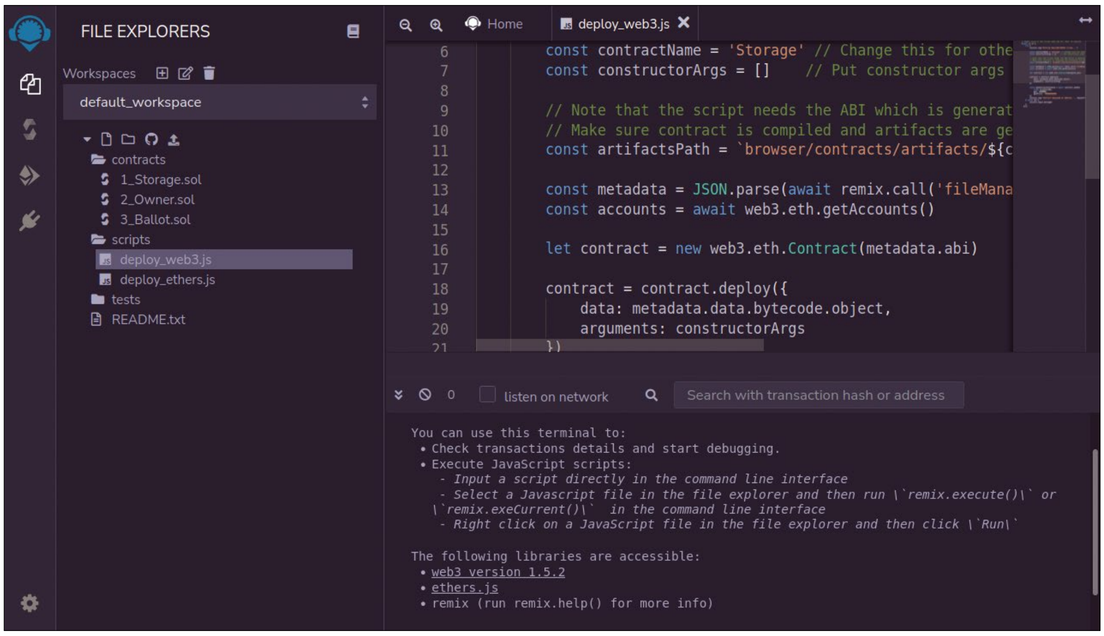
```
 2 txs (  0.123 Mg) in   7.008ms (17.583 Mg/s). #2813 [df8b1950???]                                          [5/147]
[DEBUG] 2021-12-12 12:04:19.513 [StreamLogAdapter] geth - I1212 12:04:19.513666 internal/ethapi/api.go:1263] Tx(ccc1da25aa10d48
134b9af3fd24eccf15939991f5048b5a9b7a91788099b2e03) to: &0000000000000000000000000000000000000020
[DEBUG] 2021-12-12 12:04:21.508 [StreamLogAdapter] geth - I1212 12:04:21.507896 internal/ethapi/api.go:1263] Tx(827da4b28e08e1a
f20d04f6c4b71b0c0703bd5f60a23bea53b8b1f963c61de58) to: &0000000000000000000000000000000000000020
[INFO ] 2021-12-12 12:04:22.390 [main] GethHttpServiceImpl - Geth wallet accounts unlocked (8 accounts)
[INFO ] 2021-12-12 12:04:22.477 [main] BlockchainInitializerTask - Loaded contract registry address 0x098da5468f564063e4c35af41
2bdc409d7a6d89c
[INFO ] 2021-12-12 12:04:22.537 [main] BlockchainInitializerTask - Storing existing wallet account balances
[INFO ] 2021-12-12 12:04:23.685 [main] GethHttpServiceImpl - Ethereum started successfully


                .%%%%%%%%.
              .%%/\%%%%/\%%
             .%%%\c "" J/%%%     .......
     %.      %%%%/ 6  6 \%%%     |||||||
     `%%.    %%%%    _  |%%%  @@@@@@@@@@@
      `%%    `%%%%(__Y__)%%'  {~*~*~*~*~}
      //      ;%%%%`\-/%%%' @@@@@@@@@@@@@@@@
     ((      /  `%%%%%%%'   {~* CAKESHOP *~}
      \\    .'         |    {     OPEN     }
       \\  /       \ | |    {~~*~~*~~*~~*~~}
        \\/         ) | |   @@@@@@@@@@@@@@@
         \         /_ | |__      __) (__
         (___________)))))))    /_______\

         CAKESHOP
         version:     0.10.0
         build id:    8d2a5ab8aaf0f895cec939772e252fc032958ed3
         build date:  2017-07-14T16:13:27.516-0400
         url:         http://172.17.0.1:8080/cakeshop/

[INFO ] 2021-12-12 12:04:23.954 [main] SpringBootApplication - Started SpringBootApplication in 113.752 seconds (JVM running fo
r 123.79)
21.10 0:-- 1:[tmux]*                                        1h16m 2.30 4x2.0GHz 7.7G74% 110G83% 2021-12-12 12:07:27
```

Figure 5: Cakeshop terminal output: The lion with the cake is a marker so that you can see the address to point your browser.



```
pragma solidity ^0.4.9;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    // Create a new ballot with $(_numProposals) different proposals.
    function Ballot(uint8 _numProposals) {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }

    // Give $(voter) the right to vote on this ballot.
    // May only be called by $(chairperson).
    function giveRightToVote(address voter) {
        if (msg.sender != chairperson || voters[voter].voted) return;
        voters[voter].weight = 1;
    }

    // Delegate your vote to the voter $(to).
    function delegate(address to) {
        Voter sender = voters[msg.sender]; // assigns reference
        if (sender.voted) return;
        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
            to = voters[to].delegate;
        if (to == msg.sender) return;
        sender.voted = true;
        sender.delegate = to;
        Voter delegate = voters[to];
        if (delegate.voted)
            proposals[delegate.vote].voteCount += sender.weight;
        else
            delegate.weight += sender.weight;
    }

    // Give a single vote to proposal $(proposal).
    function vote(uint8 proposal) {
        Voter sender = voters[msg.sender];
        if (sender.voted || proposal >= proposals.length) return;
        sender.voted = true;
```

Figure 6: In the sandbox, you can both write the code and test what happens when you use the functions.



```
    const contractName = 'Storage' // Change this for othe
    const constructorArgs = []    // Put constructor args

    // Note that the script needs the ABI which is generat
    // Make sure contract is compiled and artifacts are ge
    const artifactsPath = `browser/contracts/artifacts/${c

    const metadata = JSON.parse(await remix.call('fileMana
    const accounts = await web3.eth.getAccounts()

    let contract = new web3.eth.Contract(metadata.abi)

    contract = contract.deploy({
        data: metadata.data.bytecode.object,
        arguments: constructorArgs
    })
```

```
You can use this terminal to:
 • Check transactions details and start debugging.
 • Execute JavaScript scripts:
   - Input a script directly in the command line interface
   - Select a Javascript file in the file explorer and then run \`remix.execute()\` or
 \`remix.exeCurrent()\`  in the command line interface
   - Right click on a JavaScript file in the file explorer and then click \`Run\`

The following libraries are accessible:
 • web3 version 1.5.2
 • ethers.js
 • remix (run remix.help() for more info)
```

Figure 7: Remix IDE: The sample code above deploys Web3 components.

The first function stores the value in `number`, and the second function retrieves it.

Listing 1: A Smart Contract

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */



contract Storage {

    uint256 number;

    /**
     * @dev Store value in variable
     * @param num value to store
     */
    function store(uint256 num) public {
        number = num;
    }

    /**
     * @dev Return value
     * @return value of 'number'
     */
    function retrieve() public view returns (uint256){
        return number;
    }
}
```

ETH2 Is Coming Soon

Ethereum has an update coming soon; it goes to version 2.0. The most notable difference is that it goes to a Proof of Stake system instead of Proof of Work system. What that means is that you can use a small computer as a node and help the network validate the blocks. Earlier you had to have a beast of a machine to ever validate and get rewarded.

This sounds great for the little guy, but you now have to have 32 ether to participate. That amount is staked; if you behave well the amount will go up because of rewards. If you misbehave, you can be punished and lose funds. This is to ensure that nobody tries to create fake blocks and ruin consensus.

Conclusion

Ethereum is growing into a formidable force in finance and tech. Being able to use different wallets and services can help you handle your own finances – and maybe even publish your own art.

I should add that the power of Ethereum comes with the need for caution. Smart contracts are visible on the blockchain, and a bug or security hole in the code could lead to exploitation. Back in 2016, nearly $50 million in ether was drained from Ethereum accounts in this kind of attack, although the developers were able to claw the money back through a hard fork of the Ethereum blockchain. Several updates have improved security since that time, but be aware of the need to test your code thoroughly and use secure programming techniques.

Web 3.0 will be based on digital currencies and their surrounding technologies. Hopefully Ethereum will improve the web experience for most users and help balance the power on the web. ■■■

Info

[1] Ethereum: *https://ethereum.org/*

[2] NFTs: *https://en.wikipedia.org/wiki/Non-fungible_token*

[3] Ethereum's Merkle Patricia trees: *https://rockwaterweb.com/ethereum-merkle-patricia-trees-javascript-tutorial/,*

[4] Ethereum Ropsten Testnet Faucet: *https://testnet.help/en/ethfaucet/ropsten.*

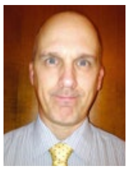
[5] Solidity: *https://docs.soliditylang.org/en/v0.8.11/*

[6] Cakeshop on GitHub: *https://github.com/ConsenSys/cakeshop/releases*

[7] Cakeshop home page: *https://opensourcelibs.com/lib/cakeshop*

[8] Remix IDE: *https://remix.ethereum.org/*

Author



Mats Tage Axelsson has a set of modes that he still has not found the keyboard shortcut to change.

Deleting metadata from files

Housekeeping

Digital camera images often reveal personal information embedded in metadata. Several Linux tools let you remove unwanted metadata to help preserve your privacy. *By Erik Bärwaldt*

The progressive digitalization of all areas of life makes it difficult to cope with the growing flood of data. Many users like to share photos taken with their smartphones or webcams, music and video files, and even PDF documents on the Internet. However, all of these file formats contain metadata that can be read with various programs and often reveals private information about the file's creator. Before you publish files on the Internet, you should first check for unwanted metadata and remove sensitive information to protect your privacy. This article looks at some Linux tools that will help you delete metadata from your files at the push of a button.

The Purpose of Metadata

Metadata is usually generated on the basis of defined specifications and intended to record certain characteristics of the files concerned. For example, in the case of image and video files, this metadata includes the camera model and the photo's technical settings. Often the metadata also includes time and date stamps and – if supported and activated on the hardware side – the GPS coordinates where the photo was taken. For all file formats, the author can also be listed in the metadata to make it easier to prove copyright infringements. In addition, the metadata can be used to keyword the files in order to categorize them and make them easier to find in applications such as image collections.

Standards

For video and image files, Exif, IPTC, and XMP are the standards for generating metadata. The respective data can be found in fixed data fields in the files' headers.

The International Press Telecommunications Council – Information Interchange Model (IPTC-IIM) specification [1] was defined as early as 1991 and can be used for all digital media.

More targeted to technical details, the much younger Exchangeable Image File Format (Exif) standard [2] is used on almost all common smartphones and digital cameras to provide images with metadata. Exif's last specification dates from 2010 and was last revised in 2019.

The Extensible Metadata Platform (XMP) [3] container format, similar to Exif, primarily maps the metadata for image files. However, XMP, which was developed by Adobe about 20 years ago, can also store free text and is therefore popular with applications such as Lightroom or darktable. All Adobe products support XMP.

Reading and Editing

Reading metadata does not require any special application software: The file managers of all common desktop interfaces can handle the existing standards. The file managers usually show some of the identified metadata when you view a file, for example, in an image viewer. In many cases, you can also display the most important metadata in a context menu in the respective file manager's Properties dialog (Figure 1). However, to edit metadata, you need special applications that usually focus on one of the existing specifications.

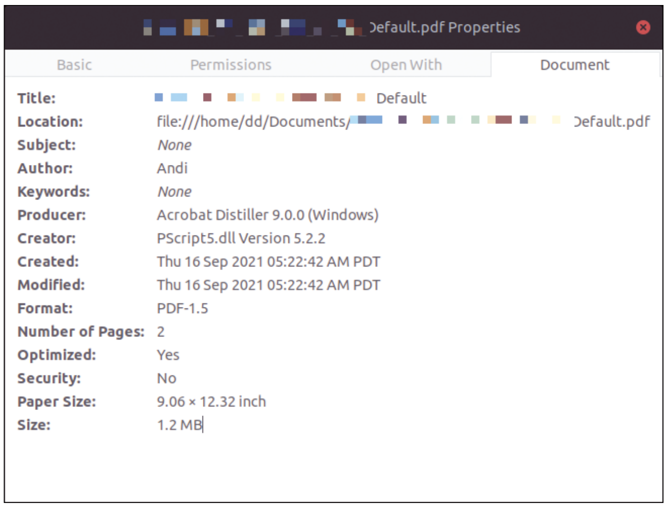
ExifTool

The cross-platform ExifTool [4] has become the quasi-standard for editing metadata. The command-line tool has a powerful command set and can handle a wide variety of image and video formats. ExifTool's developers provide an easily understandable table [5] that lists the editing options for each format.

ExifTool can be found in the repositories of most common Linux distributions; consequently, it can usually conveniently be set up via the respective package manager. After installing ExifTool, in the simplest case, you can display a file's existing Exif data by entering the command:

```
exiftool FILE
```

Photo by Robert Linder on Unsplash



[redacted] Default.pdf Properties

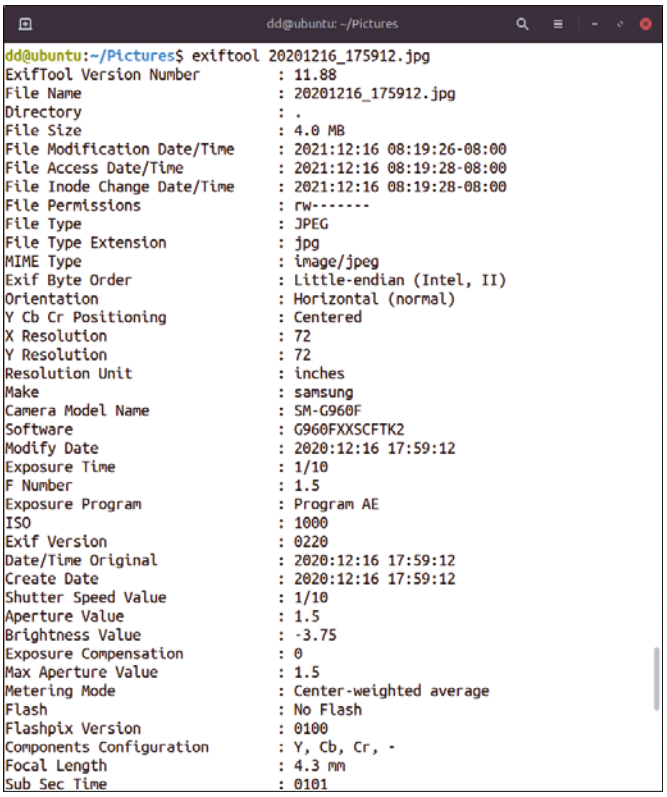
Basic | Permissions | Open With | Document

Title:	[redacted] Default
Location:	file:///home/dd/Documents/[redacted] Default.pdf
Subject:	None
Author:	Andi
Keywords:	None
Producer:	Acrobat Distiller 9.0.0 (Windows)
Creator:	PScript5.dll Version 5.2.2
Created:	Thu 16 Sep 2021 05:22:42 AM PDT
Modified:	Thu 16 Sep 2021 05:22:42 AM PDT
Format:	PDF-1.5
Number of Pages:	2
Optimized:	Yes
Security:	No
Paper Size:	9.06 × 12.32 inch
Size:	1.2 MB

Figure 1: A conventional file manager can reveal information about a wide variety of file formats on Linux.

Listing 1: Typical ExifTool Commands

```
01 $ exiftool --common FILE FILE.txt
02 $ exiftool -EXIF TAG='TAG CONTENT' FILE
03 $ exiftool --EXIF TAG FILE
04 $ exiftool -all=FILE
```



```
dd@ubuntu: ~/Pictures
dd@ubuntu:~/Pictures$ exiftool 20201216_175912.jpg
ExifTool Version Number         : 11.88
File Name                       : 20201216_175912.jpg
Directory                       : .
File Size                       : 4.0 MB
File Modification Date/Time     : 2021:12:16 08:19:26-08:00
File Access Date/Time           : 2021:12:16 08:19:28-08:00
File Inode Change Date/Time     : 2021:12:16 08:19:28-08:00
File Permissions                : rw-------
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
Exif Byte Order                 : Little-endian (Intel, II)
Orientation                     : Horizontal (normal)
Y Cb Cr Positioning             : Centered
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Make                            : samsung
Camera Model Name               : SM-G960F
Software                        : G960FXXSCFTK2
Modify Date                     : 2020:12:16 17:59:12
Exposure Time                   : 1/10
F Number                        : 1.5
Exposure Program                : Program AE
ISO                             : 1000
Exif Version                    : 0220
Date/Time Original              : 2020:12:16 17:59:12
Create Date                     : 2020:12:16 17:59:12
Shutter Speed Value             : 1/10
Aperture Value                  : 1.5
Brightness Value                : -3.75
Exposure Compensation           : 0
Max Aperture Value              : 1.5
Metering Mode                   : Center-weighted average
Flash                           : No Flash
Flashpix Version                : 0100
Components Configuration        : Y, Cb, Cr, -
Focal Length                    : 4.3 mm
Sub Sec Time                    : 0101
```

Figure 2: Working from the command line, ExifTool lists existing metadata in detail.

The application takes all the existing metadata into account, so the output is usually quite long and confusing (Figure 2).

To save the most important metadata separately, redirect the output to a text file (Listing 1, line 1). In addition to the metadata, the Exif tags are also included in the file. To edit individual Exif tags, use the command from line 2 and then verify the modified tag with the command from line 3. As a result, you get a single line with the desired Exif tag and its value. If you want to delete all the tags to protect your privacy prior to publishing an image on the Internet, enter the command from the line 4 at the prompt.

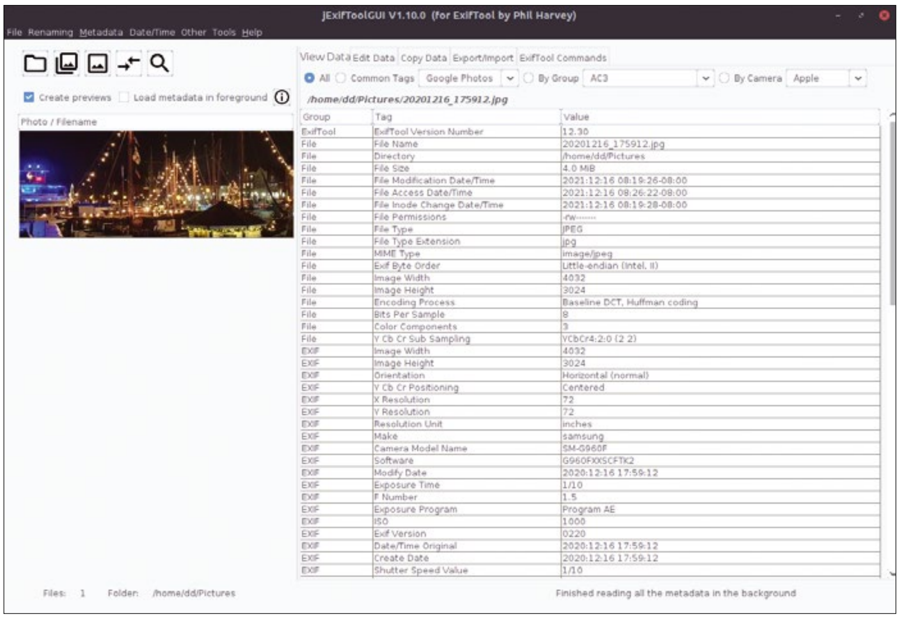
jExifToolGUI

Due to the complexity of the editing options and the large volume of metadata, ExifTool requires a longer learning curve. It is far easier to edit metadata using the jExifToolGUI [6], which provides a graphical user interface (GUI). The Java-based application requires the installation of ExifTool and version 11 of the Java Runtime Environment, which can be easily installed from the repositories of almost all popular distributions.

After a somewhat slow start, jExifToolGUI comes up with a clear-cut interface. On the left, below the menubar and buttonbar, you will find a pane that displays small thumbnails of the files discovered by jExifToolGUI. To the right of this pane, you can view and edit the metadata of the selected files using the tabs at the top of this pane.

To get started, click on the folder icon in the top left corner of the window and select the desired folder in the file manager that then opens. jExifToolGUI then reads the files it discovers and displays them one below the other as thumbnails in the pane on the left. After clicking on one of the files, the associated metadata appears in a table on the right in the *View Data* tab (Figure 3).

Using the *Edit Data* tab, you can then modify the existing data. A second tab bar, which categorizes the corresponding data, lets you to reach the data quickly



jExifToolGUI V1.10.0 (for ExifTool by Phil Harvey)

File Renaming Metadata Date/Time Other Tools Help

☑ Create previews ☐ Load metadata in foreground

Photo / Filename

View Data | Edit Data | Copy Data | Export/Import | ExifTool Commands

● All ○ Common Tags Google Photos ○ By Group AC3 ○ By Camera Apple

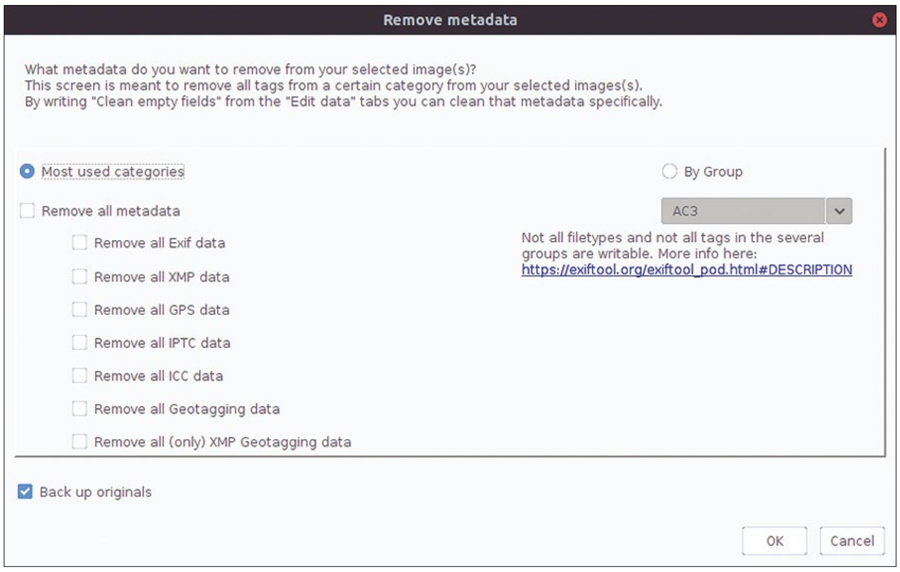
/home/dd/Pictures/20201216_175912.jpg

Group	Tag	Value
ExifTool	ExifTool Version Number	12.30
File	File Name	20201216_175912.jpg
File	Directory	/home/dd/Pictures
File	File Size	4.0 MiB
File	File Modification Date/Time	2021:12:16 08:19:26-08:00
File	File Access Date/Time	2021:12:16 08:26:22-08:00
File	File Inode Change Date/Time	2021:12:16 08:19:28-08:00
File	File Permissions	-rw-------
File	File Type	JPEG
File	File Type Extension	jpg
File	MIME Type	image/jpeg
File	Exif Byte Order	Little-endian (Intel, II)
File	Image Width	4032
File	Image Height	3024
File	Encoding Process	Baseline DCT, Huffman coding
File	Bits Per Sample	8
File	Color Components	3
File	Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
EXIF	Image Width	4032
EXIF	Image Height	3024
EXIF	Orientation	Horizontal (normal)
EXIF	Y Cb Cr Positioning	Centered
EXIF	X Resolution	72
EXIF	Y Resolution	72
EXIF	Resolution Unit	inches
EXIF	Make	samsung
EXIF	Camera Model Name	SM-G960F
EXIF	Software	G960FXXSCFTK2
EXIF	Modify Date	2020:12:16 17:59:12
EXIF	Exposure Time	1/10
EXIF	F Number	1.5
EXIF	Exposure Program	Program AE
EXIF	ISO	1000
EXIF	Exif Version	0220
EXIF	Date/Time Original	2020:12:16 17:59:12
EXIF	Create Date	2020:12:16 17:59:12
EXIF	Shutter Speed Value	1/10

Files: 1 Folder: /home/dd/Pictures

Finished reading all the metadata in the background

Figure 3: jExifToolGUI lets users view and edit metadata in an easy-to-use GUI.



Remove metadata

What metadata do you want to remove from your selected image(s)?
This screen is meant to remove all tags from a certain category from your selected images(s).
By writing "Clean empty fields" from the "Edit data" tabs you can clean that metadata specifically.

[✓ Most used categories] By Group

AC3

Remove all metadata
- Remove all Exif data
- Remove all XMP data
- Remove all GPS data
- Remove all IPTC data
- Remove all ICC data
- Remove all Geotagging data
- Remove all (only) XMP Geotagging data

Not all filetypes and not all tags in the several groups are writable. More info here:
https://exiftool.org/exiftool_pod.html#DESCRIPTION

[✓] Back up originals

OK Cancel

Figure 4: Existing metadata is quickly deleted in jExifToolGUI if necessary.

without having to trudge through a long, confusing table of Exif tags and their values. The Exif tags belonging to the respective category appear one below the other on the individual tabs; you can enter the modified values in corresponding input fields. To the right, you click checkboxes to save the new values; by default, jExifToolGUI checks all the boxes for saving the values.

After completing the modifications, transfer the new metadata to the respective image(s) in each tab by clicking the *Copy to selected image(s) and save* button at the bottom of the active dialog. jExifToolGUI will then transfer the modified data to the correct location in the metadata inventory. If necessary, you can also simply copy existing data to the appropriate tab category by clicking *Copy from selected images* at the bottom of the relevant editing segment after selecting an image. By selecting this option, you must reread the data in each category for the selected images.

Metadata | *Remove metadata* saves you from having to laboriously go through the individual editing dialogs one by one to remove metadata. Clicking on it opens a new dialog in which you select the categories in question by checking the matching options (Figure 4). At the same time, the default *Back up originals* option ensures that backups of the original data are preserved.

After clicking *OK* and confirming the prompt, jExifToolGUI removes the metadata from the image header. If you then reselect the image and reread the metadata using the *Copy from selected images* button, you will see empty fields for the corresponding Exif tags.

ExifCleaner

ExifCleaner [7] also relies on ExifTool as the basis for reading and removing Exif data. Due to its simple user interface, ExifCleaner is best suited for users who only occasionally want to remove metadata from images. The program cannot edit metadata.

You will find RPM and DEB binary packages for ExifCleaner on GitHub, as well as a generic AppImage package. The source code is also available on GitHub.

ExifCleaner opens a spartan-looking empty window with only one menubar. Drag the image files whose metadata you want to remove into the empty space. In two columns, the tool shows how many Exif values it has found in each image file and how many values will remain after the removal process (Figure 5). You can only see the number of read-only values that cannot be removed, but not which Exif tags the values belong to.

ExifCleaner automatically starts removing the metadata immediately after loading the files. Since the tool does not back up the original files, the deleted metadata cannot be reconstructed. Take care when selecting and transferring image files to ExifCleaner to avoid data loss.

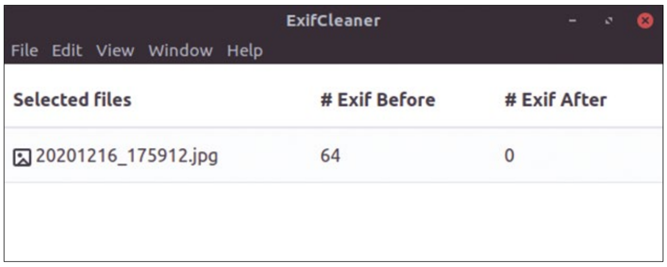
Contrary to what the program name suggests, ExifCleaner does not just manipulate image files, but also video formats such as M4A, MOV, QT, and MP4. In addition, ExifCleaner removes all metadata except for read-only metadata (which cannot be deleted) from PDF files.

jhead

You can use `jhead` [8], a small command-line program, to modify the Exif headers in JPEG files. Due to the software's considerable number of parameters, `jhead` requires a longer training period depending on your intended use. Because `jhead` can only handle JPEGs, you will get an error message if you call a file in any other format.

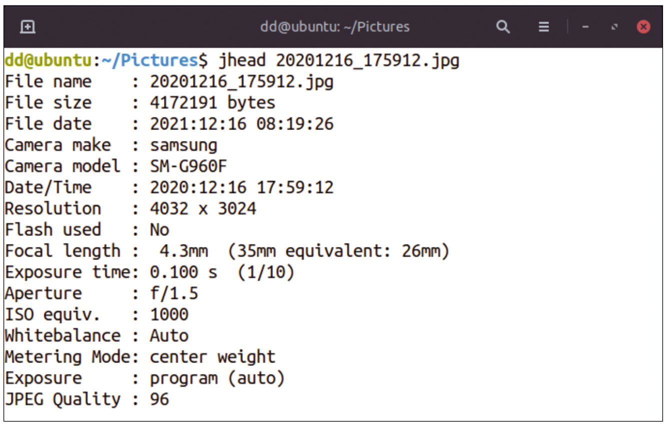
To display an image file's metadata, use the command:

```
jhead FILE
```



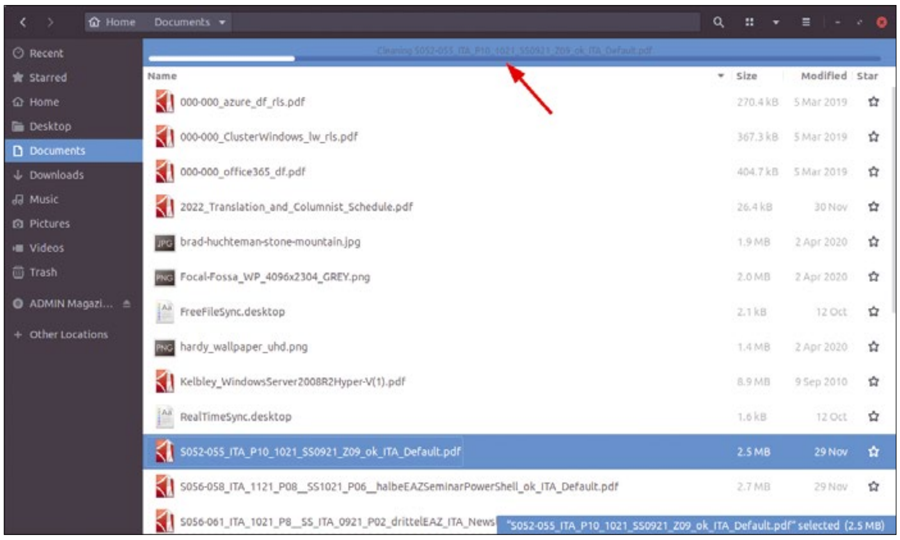
Selected files	# Exif Before	# Exif After
20201216_175912.jpg	64	0

Figure 5: ExifCleaner is a plain vanilla metadata removal tool.



```
dd@ubuntu:~/Pictures$ jhead 20201216_175912.jpg
File name    : 20201216_175912.jpg
File size    : 4172191 bytes
File date    : 2021:12:16 08:19:26
Camera make  : samsung
Camera model : SM-G960F
Date/Time    : 2020:12:16 17:59:12
Resolution   : 4032 x 3024
Flash used   : No
Focal length :  4.3mm  (35mm equivalent: 26mm)
Exposure time: 0.100 s  (1/10)
Aperture     : f/1.5
ISO equiv.   : 1000
Whitebalance : Auto
Metering Mode: center weight
Exposure     : program (auto)
JPEG Quality : 96
```

Figure 6: `jhead` deletes or modifies metadata in JPEG files only.



Cleaning S052-055_ITA_P10_1021_SS0921_Z09_ok_ITA_Default.pdf

Name	Size	Modified	Star
000-000_azure_df_rls.pdf	270.4 kB	5 Mar 2019	☆
000-000_ClusterWindows_lw_rls.pdf	367.3 kB	5 Mar 2019	☆
000-000_office365_df.pdf	404.7 kB	5 Mar 2019	☆
2022_Translation_and_Columnist_Schedule.pdf	26.4 kB	30 Nov	☆
brad-huchteman-stone-mountain.jpg	1.9 MB	2 Apr 2020	☆
Focal-Fossa_WP_4096x2304_GREY.png	2.0 MB	2 Apr 2020	☆
FreeFileSync.desktop	2.1 kB	12 Oct	☆
hardy_wallpaper_uhd.png	1.4 MB	2 Apr 2020	☆
Kelbley_WindowsServer2008R2Hyper-V(1).pdf	8.9 MB	9 Sep 2010	☆
RealTimeSync.desktop	1.6 kB	12 Oct	☆
S052-055_ITA_P10_1021_SS0921_Z09_ok_ITA_Default.pdf	2.5 MB	29 Nov	☆
S056-058_ITA_1121_P08__SS1021_P06__halbeEAZSeminarPowerShell_ok_ITA_Default.pdf	2.7 MB	29 Nov	☆
S056-061_ITA_1021_P8__SS_ITA_0921_P02_drittelEAZ_ITA_News			

"S052-055_ITA_P10_1021_SS0921_Z09_ok_ITA_Default.pdf" selected (2.5 MB)

Figure 7: **You can use mat2 in the Gnome file manager Nautilus via a plugin.**

The application then lists the individual Exif tags in the terminal (Figure 6). To change the tags based on the parameters supported by the software, use the command:

```
jhead PARAMETER FILE
```

The individual parameters are revealed by calling `jhead -h`. To change the same Exif tag for multiple files located in a directory, you can also specify the file names using wildcards, resulting in `jhead` doing the job in one fell swoop.

mat2

A classic tool for removing metadata from file collections, the Metadata Anonymisation Toolkit 2 (mat2) [9] not only supports images and audio and video files, but it also supports PDF and Office file formats. This makes mat2 particularly suitable for users in sensitive areas, such as journalists or lawyers.

The command-line tool offers numerous options and parameters, which require some familiarization. However, a plugin for the Gnome file manager Nautilus lets you clean up groups of files at the command line even without knowing the syntax (Figure 7). To do this, you simply select the files to be cleaned and then right-click to call up a context menu in which you click the *Remove metadata* option.

Using the plugin, you can select files with a wide variety of formats; mat2 identifies the format and applies the correct routines to remove the metadata. During cleanup, the plugin also displays a progress bar at the top of the program window. In addition to the original file, which remains unprocessed, the cleaned file, which has the same name with `cleaned` appended, is created in the working directory.

The Tails security distribution ships mat2 with the Nautilus plugin pre-configured.

Metadata Cleaner

Only a few distributions currently have Metadata Cleaner [10] in their repositories. Native binaries are available for Arch Linux, Debian, and Ubuntu and their descendants; a Flatpak and the Python source code are also available.

After installation and first startup, Metadata Cleaner loads with a visually modern and functionally spartan window: Metadata Cleaner has no conventional menubar or buttonbar. Instead, the GTK-based application follows the Gnome operating principle and integrates all controls of interest in the titlebar (Figure 8).

The *Add Files* button in the top left corner takes you to a file manager where you can select the image files to be processed. Metadata Cleaner then displays them in a list, with the number of identified Exif tags highlighted in red in a column on the right. Clicking on the rightmost arrow after each file icon opens a detailed display of the metadata embedded in the respective image file. The arrow pointing to the left in the titlebar takes you back to the file display.

To remove the metadata from all of the loaded files, simply press the big red *Clean* button bottom right in the program window. Metadata Cleaner will suggest creating a backup copy and then it will delete all metadata it finds without further ado. The application does not automatically create any backup copies of the originals, so the metadata cannot be reconstructed later.

If you click on the gear icon to the left of the *Clean* button, a small dialog opens in which you can switch on a *Superficial cleanup* via a slider. This mode does not completely remove the metadata from the respective file, but tries to preserve functionally relevant metadata. The *Superficial cleanup* is best suited for file formats such as PDF, where you can also make changes. Modifications such as text changes in a PDF or image compression may be controlled by the metadata and will be lost if the meta tags are missing. Consequently, you should not delete all the metadata for these types of file formats. The slider effects all files stored in the program window. Cleaning up the files takes some time, depending on the number.

digiKam

With digiKam [11], which is part of the KDE Plasma library, you can view metadata with the push of a button and edit the data in a convenient editor. You'll find digiKam in the repositories of all of the popular distributions, and

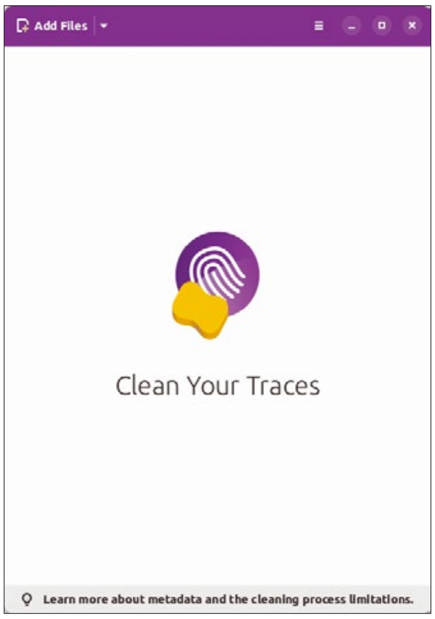

Add Files
Clean Your Traces
Learn more about metadata and the cleaning process limitations.

Figure 8: **Metadata Cleaner stands out due to its brightly colored interface.**

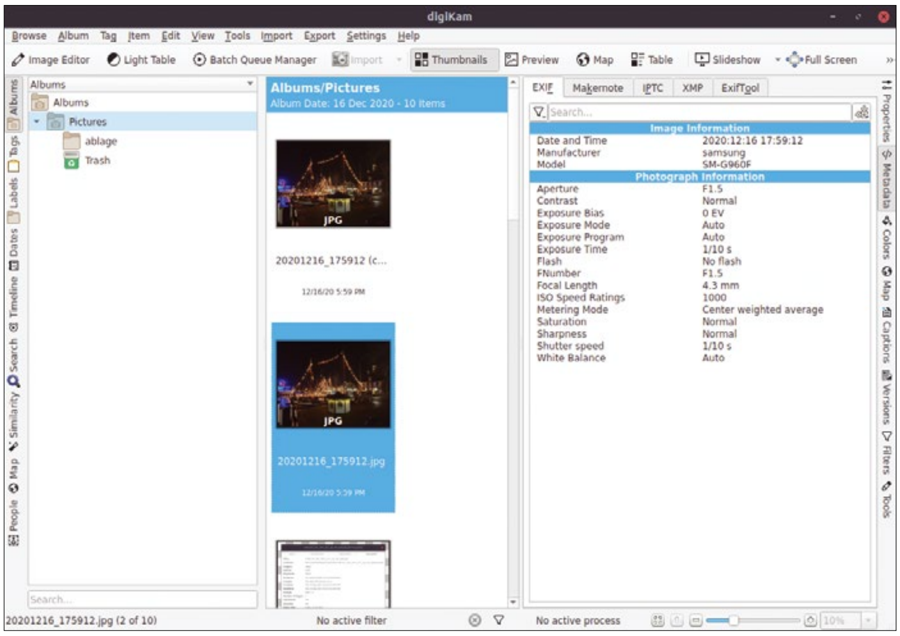

digiKam
Browse Album Tag Item Edit View Tools Import Export Settings Help
Image Editor
Light Table
Batch Queue Manager
Import
Thumbnails
Preview
Map
Table
Slideshow
Full Screen
Albums
Pictures
ablage
Trash
Albums/Pictures
Album Date: 16 Dec 2020 - 10 items
20201216_175912.jpg
EXIF Makernote IPTC XMP ExifTool
Properties Metadata Colors Map Captions Versions Filters Tools
20201216_175912.jpg (2 of 10)
No active filter
No active process

Image Information	
Date and Time	2020:12:16 17:59:12
Manufacturer	samsung
Model	SM-G960F
Photograph Information	
Aperture	F1.5
Contrast	Normal
Exposure Bias	0 EV
Exposure Mode	Auto
Exposure Program	Auto
Exposure Time	1/10 s
Flash	No flash
FNumber	F1.5
Focal Length	4.3 mm
ISO Speed Ratings	1000
Metering Mode	Center weighted average
Saturation	Normal
Sharpness	Normal
Shutter speed	1/10 s
White Balance	Auto

Figure 9: A heavyweight among image editing programs, digiKam has a fairly cluttered appearance at first.

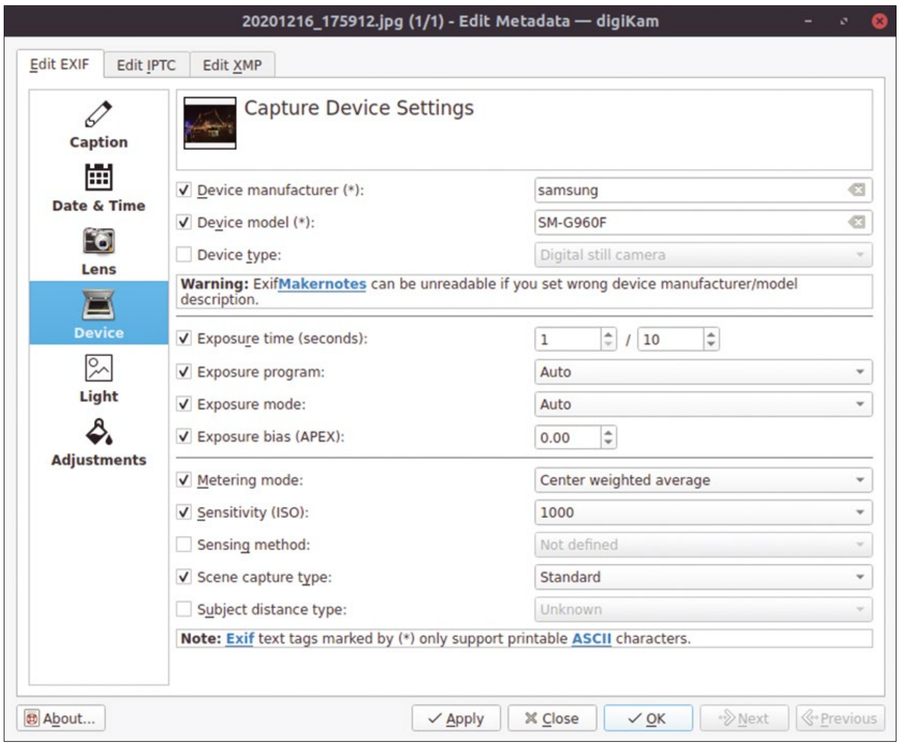
you can use digiKam with any desktop environment. When you launch digiKam for the first time, a multi-step wizard helps out with the basic configuration, which includes paths and settings for the software, as well the link to a database back end.

The program window that appears after the initial configuration has a conventional menubar and buttonbar for frequently used functions and three main panes below (Figure 9). On the left, the system's folder hierarchy is displayed as a tree view; in the center, you can see thumbnails of the image files that reside in the current folder. On the far right, digiKam displays information about the selected image.

If you select the *Metadata* tab in the unusual but space-saving vertical tab bar on the far right, the data from the headers of the active image appear in the information pane on the right. digiKam also divides this pane into several tabs. By default, it displays the Exif tags. In addition, the *IPTC* and *XMP* tabs will show the metadata for these specifications if available.

To edit the metadata in the active image, click *Item* in the menubar and select *Edit Metadata* from the context menu that opens. A separate dialog now lists all metadata sorted by tags and lets you modify the field content (Figure 10). You can also exclude or include individual fields by checking or unchecking the boxes to the left. On the left side of the dialog box, you can choose between several categories that group similar settings. At the top of the workspace, there are additional tabs that let you switch between the different metadata standards.

The dialogs for the IPTC and XMP standard primarily contain comment fields, a copyright note, and keywords for keyword tagging. The categories in the left sidebar change to reflect the options available for each standard. Once you have made your changes, first click *Apply* bottom left in the buttonbar and then *OK* to close the dialog. You are then returned to the digiKam main window.



Capture Device Settings

✓ Device manufacturer (*):	samsung
✓ Device model (*):	SM-G960F
☐ Device type:	Digital still camera

Warning: ExifMakernotes can be unreadable if you set wrong device manufacturer/model description.

✓ Exposure time (seconds):	1 / 10
✓ Exposure program:	Auto
✓ Exposure mode:	Auto
✓ Exposure bias (APEX):	0.00
✓ Metering mode:	Center weighted average
✓ Sensitivity (ISO):	1000
☐ Sensing method:	Not defined
✓ Scene capture type:	Standard
☐ Subject distance type:	Unknown

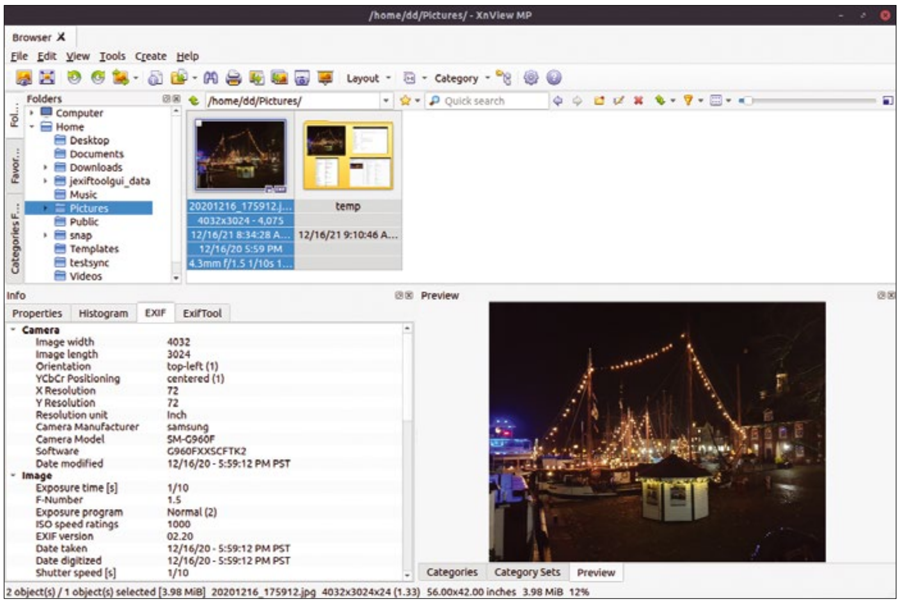
Note: Exif text tags marked by (*) only support printable ASCII characters.

Figure 10: You can edit metadata in digiKam conveniently as field content.

XnView MP

The commercial XnView MP [12] tool, which is distributed as freeware for private and educational use, is a cross-platform image viewer with numerous image editing functions. XnView MP supports more than 500 image formats and is available for download for Linux from the vendor's website as a TGZ archive, a DEB binary package, and as an AppImage. The DEB package integrates seamlessly into Ubuntu and its derivatives and also creates an entry in the respective desktop's menu tree.

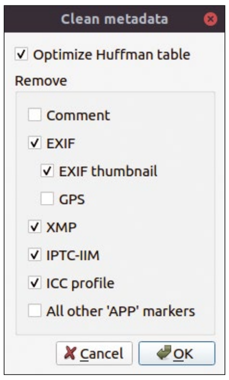
After starting XnView MP, a clear-cut, tab-based interface opens (Figure 11); it has four panes, a menubar, and a buttonbar. The top left pane contains a tree view of the system's folder hierarchy; in a large pane on the right, XnView MP displays the images from the current folder as previews. In the bottom right pane, you'll find a horizontal category list for tagging the images, as well as an info segment in the bottom left pane that displays the matching metadata in the *EXIF*, *IPTC-IIM*, and *XMP* tabs. However, if the current image



Camera
Image width 4032
Image length 3024
Orientation top-left (1)
YCbCr Positioning centered (1)
X Resolution 72
Y Resolution 72
Resolution unit Inch
Camera Manufacturer samsung
Camera Model SM-G960F
Software G960FXXSCFTK2
Date modified 12/16/20 - 5:59:12 PM PST
Image
Exposure time [s] 1/10
F-Number 1.5
Exposure program Normal (2)
ISO speed ratings 1000
EXIF version 02.20
Date taken 12/16/20 - 5:59:12 PM PST
Date digitized 12/16/20 - 5:59:12 PM PST
Shutter speed [s] 1/10

2 object(s) / 1 object(s) selected [3.98 MiB] 20201216_175912.jpg 4032x3024x24 (1.33) 56.00x42.00 inches 3.98 MiB 12%

Figure 11: XnView MP also offers numerous image processing functions.



☑ Optimize Huffman table
Remove
☐ Comment
☑ EXIF
☑ EXIF thumbnail
☐ GPS
☑ XMP
☑ IPTC-IIM
☑ ICC profile
☐ All other 'APP' markers

Figure 12: Metadata can be deleted in XnView MP with a few mouse clicks.

has no metadata for a certain standard, XnView MP does not show you a tab for that standard.

Under *Tools* | *Metadata*, you will find various options for editing metadata, sorted by the supported standards. A separate *Edit GPS data* entry also lets you change existing GPS coordinates. The *Delete* option removes existing metadata completely, and you can specify which metadata XnView MP should delete in another dialog by checking a box (Figure 12). However, XnView MP does not consider individual tags, but only the respective standards.

While XnView MP can only edit a number of selected Exif tags, it offers more advanced modifications for IPTC and XMP metadata. To modify Exif tags, go to the *Transfer* dialog under *Tools* | *Metadata*, select the individual tags from a list, and populate them with predefined values from a drop-down list.

Conclusions

Linux proves to be very flexible when editing metadata in a wide variety of file formats. While there fewer Linux applications for this tasks compared with other operating systems, the tools that are available cover all common areas of application (see Table 1). Both command-line users and users who prefer a GUI will find a solution to meet their needs.

Image editors like digiKam or XnView MP require some training, as they cover a very large range of functions, which makes their interfaces more confusing. Casual users who just want to remove metadata from their images for publication are better off using one of the point-and-click, meta tag removal-only programs. ■■■

Info

[1] IPTC-IIM: *https://en.wikipedia.org/wiki/IPTC_Information_Interchange_Model*
[2] Exif: *https://en.wikipedia.org/wiki/Exif*
[3] XMP: *https://en.wikipedia.org/wiki/Extensible_Metadata_Platform*
[4] ExifTool: *https://exiftool.org/*
[5] ExifTool editing options: *https://exiftool.org/#supported*
[6] jExifToolGUI: *https://hvdwolf.github.io/jExifToolGUI/*
[7] ExifCleaner: *https://exifcleaner.com/*
[8] jhead: *https://www.sentex.ca/~mwandel/jhead/*
[9] mat2: *https://0xacab.org/jvoisin/mat2*
[10] Metadata Cleaner: *https://flathub.org/apps/details/fr.romainvigier.MetadataCleaner*
[11] digiKam: *https://www.digikam.org/*
[12] XnView MP: *https://www.xnview.com/de/xnviewmp/*

Table 1: Metadata Editing Tools

	ExifTool	jExifToolGUI	ExifCleaner	jhead	mat2	Metadata Cleaner	digiKam	XnView MP
License	GPL	GPL	MIT	Public Domain	LGPL	GPL	GPL	Commercial
Properties								
GUI	No	Yes	Yes	No	Yes^1	Yes	Yes	Yes
Text-Based Interface	Yes	No	No	Yes	Yes	No	No	No
Cross-Platform	Yes	Yes^2	Yes	Yes	Yes	No	Yes	Yes
Supported Standards/Formats								
Exif/IPTC/XMP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multimedia (Image/Audio/Video)	Yes	Yes	Yes	JPEG	Yes	Yes	Yes	Yes
PDF Files	No	No	Yes^3	No	Yes	Yes	No	No
Office Files	No	No	No	No	Yes	No	No	No
Functions								
Removing Metadata	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Modifying Metadata	Yes	Yes	No	Yes	No	No	Yes	Yes
^{1}File Manager, ^{2}Java, ^{3}Restricted								

Package maintenance at the command line

Goody Bag

Debian Goodies lets you manage and troubleshoot packages from the command line. *By Bruce Byfield*

Many users of Debian or its derivatives are aware of only a handful of tools. When troubleshooting or searching for a package to install, most rely on the Debian web pages for each package. The more expert may use `apt-cache`. However, these tools give limited information. For more demanding needs, a better alternative is Debian Goodies [1], a collection of small scripts that allow users to probe the details of packages without going through them file by file. First written and curated by Matt Zimmerman, Debian Goodies remain little known but can be equally useful for the casual user looking for a package, a sys admin troubleshooting packages, or a maintainer building a package. Perhaps their strongest feature is that, unlike the package web pages, Debian Goodies deliver results directly to the command line, without the need to switch windows.

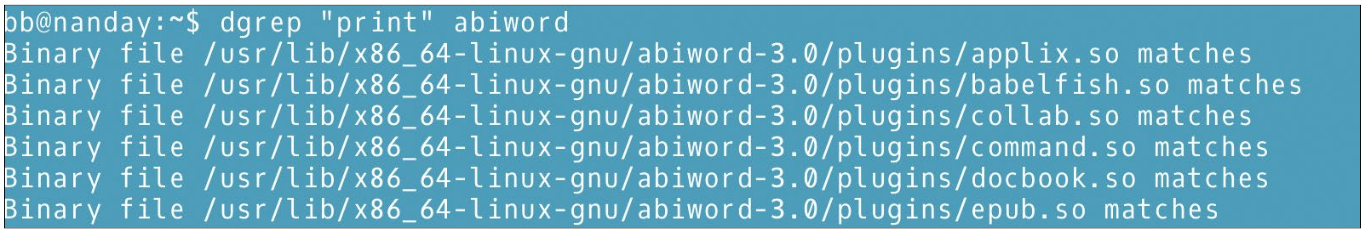
Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest Coast art (*http://brucebyfield.wordpress.com*). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

dgrep and dglob

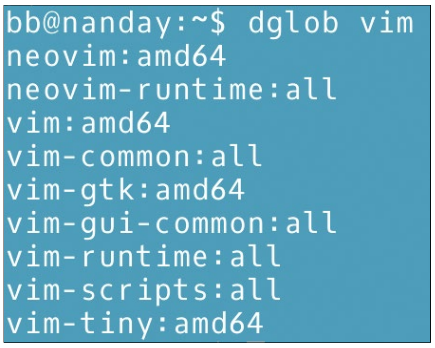
Searches can be useful for many purposes when you are troubleshooting packages. For instance, they can find what a package does, what the current package version is, or what programming language a package is written in. If all else fails, they can provide the name of the maintainer so that you can contact them. Both `dgrep` and `dglob` are in effect minor add-ons to already existing search tools, each of which deserves its own article. Here, there is only room for a general orientation.

By invoking the basic `grep` command, `dgrep` (Figure 1) searches for strings and regular expressions through installed



```
bb@nanday:~$ dgrep "print" abiword
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/applix.so matches
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/babelfish.so matches
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/collab.so matches
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/command.so matches
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/docbook.so matches
Binary file /usr/lib/x86_64-linux-gnu/abiword-3.0/plugins/epub.so matches
```

Figure 1: `dgrep` **searches packages for key strings.**



```
bb@nanday:~$ dglob vim
neovim:amd64
neovim-runtime:all
vim:amd64
vim-common:all
vim-gtk:amd64
vim-gui-common:all
vim-runtime:all
vim-scripts:all
vim-tiny:amd64
```

Figure 2: Like `dgrep`, `dglob` searches packages.

packages. Similarly, `degrep` invokes `egrep`, `dfgrep` invokes `fgrep`, and `dzgrep` invokes `zgrep`. You can limit the search by specifying the packages to search within. The available options are the same as for `grep`, except for those specific to directories (`-r`, `--recursive`, `-d recurse`, `--directories=recurse`, `-d read`, `--directories=read`) because `dgrep` skips directories. In addition, symbolic links are not searched.

Providing a slightly different approach to searches than `dgrep`, `dglob` (Figure 2) searches installed package names by default. However, adding the `-a` search broadens a search to include all available packages and omits package architectures if `-A` is added as well. By contrast, `-n` omits installed packages from the search. In addition, `-f` lists all the files in matched packages. The use of both `-a` and `-n` can be further refined by the options for `grep-dctrl(1)` and `grep-aptavail(1)`. Similarly, `-f` can be used with `apt-file`'s options.

Searches with `dglob` can also be modified with:

- `--regex` (`-rv`): Treats the entire search string as a regular expression
- `--ignore-case` (`-i`): Treats uppercase and lowercase letters the same
- `--exact-match` (`-X`): Gives only exact matches in results
- `--invert-match` (`-v`): Shows only results that do not match the search string

If these options are not enough, check the man pages of the commands that `dgrep` and `dglob` invoke.

debget

Primarily for developers rather than users, `debget` (Figure 3) lets developers download a package for study and development into their present working directory. A particular version of a package can be downloaded by adding `=VERSION` to the end of the package name.

dpigs

Similar to the `top` command, `dpigs` (Figure 4) lists the packages that take up the most space (instead of listing running applications). Like `top`, `dpigs` lists the largest memory hogs first. You can limit the number of items displayed with `--lines=NUMBER` (`-n=NUMBER`). Instead of package names, it can list package files with `--status=FILES` (`-s=FILES`). More detailed information about files can be displayed with `--human-readable` (`-H`).

debman

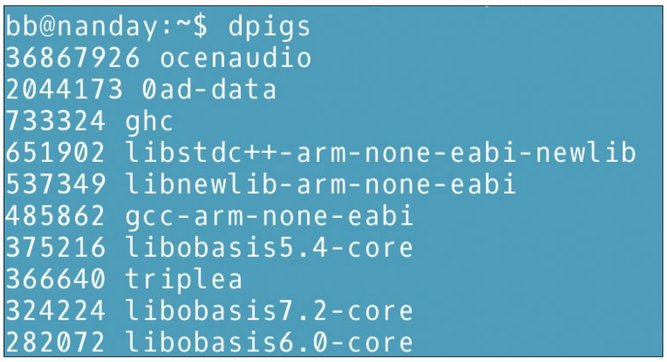
You can use `debman` (Figure 5) to quickly locate and display the man page associated with a particular package. The package is followed by the version, or, if you do not want a specific version, by a repetition of the package name. Any local environment variable like `$MANPATH` is ignored. Instead, the specified package is extracted to a temporary directory and used to retrieve its man pages. With the `-p` option, the named package will be downloaded from the repositories; the `-f` option downloads the local `.deb` file.

debmany

Using `debmany` (Figure 6), you can create a list of man pages associated with a package and then select and display them one at a time. You can choose to display a page in a viewer of your choice that can read files with a `.gz` extension using the `-k` option in KDE's Plasma, the `-g` option in Gnome, or the `-x` option in Xfce, Gnome, or Plasma. Another viewer can also be set with `-m VIEWER`. Similarly, although English is the default language, another language can be set using `-l` followed by a standard two-letter abbreviation such as `fr` for French or `de` for German. With any of these options, you can also use `-L LENGTH` to set a length limit specified in `K` (kilobytes), `M` (megabytes), `G` (gigabytes), or `T` (terabytes). Although, practically speaking, being text, man pages are short enough that the last three set no practical limit.

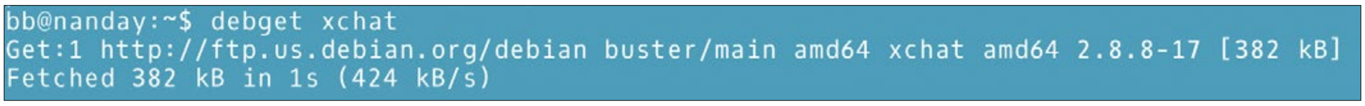
check-enhancements

The `check-enhancements` script (Figure 7) lists any packages that add functionality to a package but are not required to run it. This relationship is indicated by the fact that, if the `--verbose` option is used, results begin with "Could be enhanced by." The other options are `--installed-packages` (`-installed-packages`, `-ip`, `--ip`), which displays the enhancements for installed packages, and `--installed-enhancements` (`-installed-enhancements`, `-ie`, `--ie`), which displays results by enhancements. If no package or enhancement is entered, then the results for every package or enhancement are listed, a process that can take some time and may well be pointless, because the majority of packages on a typical system do not have enhancements. If an enhancement is available but uninstalled, a possible installation candidate is given.



```
bb@nanday:~$ dpigs
36867926 ocenaudio
2044173 0ad-data
733324 ghc
651902 libstdc++-arm-none-eabi-newlib
537349 libnewlib-arm-none-eabi
485862 gcc-arm-none-eabi
375216 libobasis5.4-core
366640 triplea
324224 libobasis7.2-core
282072 libobasis6.0-core
```

Figure 4: Just as `top` shows the largest active processes, `dpigs` shows the packages that take up the most diskspace.



```
bb@nanday:~$ debget xchat
Get:1 http://ftp.us.debian.org/debian buster/main amd64 xchat amd64 2.8.8-17 [382 kB]
Fetched 382 kB in 1s (424 kB/s)
```

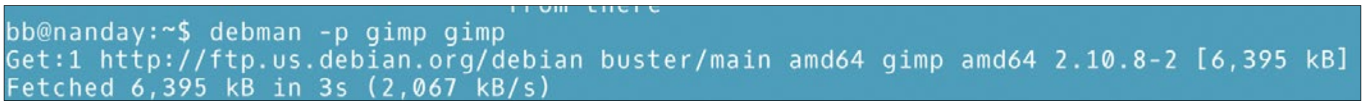
Figure 3: With `debget`, you can download a package to a local directory.

popbugs

For some users, the Debian Popularity Contest (*popularity-contest*) is a controversial package. The *popularity-contest* package is usually installed during the installation of a Debian system and sets up a regular cronjob to report on package usage to the project from which user statistics are tabulated. This information is used to decide which packages go into the next release. However, some users consider it a violation of privacy, which is why the installer gives the option of installing it. One argument in favor of *popularity-contest* is `popbugs` (Figure 8). This script correlates the *popularity-contest* data from your system with the list of critical bugs from the Debian bug-tracking system. The results are a list of critical bugs customized for your system, which can save you scanning through a complete list. If the log is enabled but has collected no data, the generalized critical bug page is shown. By default, the results are displayed in a web browser, but the option `--output=FILE` writes the results instead to a file. If you decide this comparison could be useful but did not enable *popularity-contest* during installation, you can install it at any time like any other package.

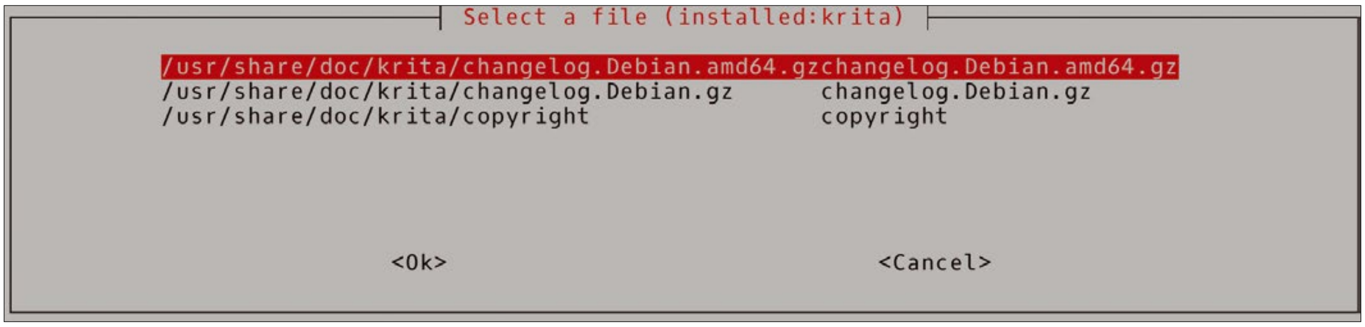
which-pkg-broke

If an attempted installation of a package results in broken packages, `which-pkg-broke` can be a useful tool for



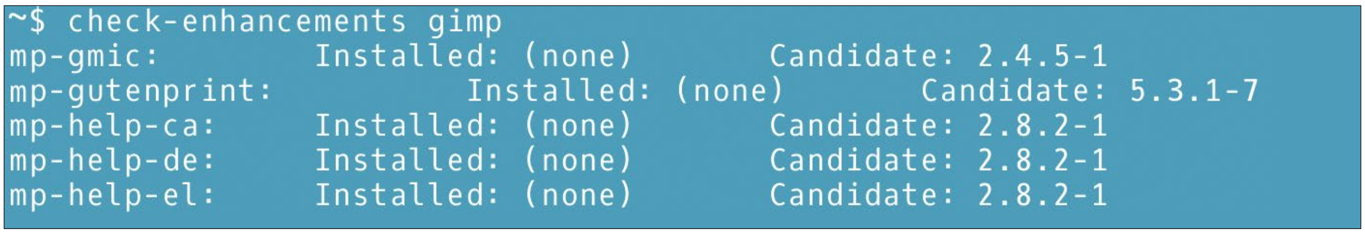
```
bb@nanday:~$ debman -p gimp gimp
Get:1 http://ftp.us.debian.org/debian buster/main amd64 gimp amd64 2.10.8-2 [6,395 kB]
Fetched 6,395 kB in 3s (2,067 kB/s)
```

Figure 5: Use `debman` to locate and display a package's man page.



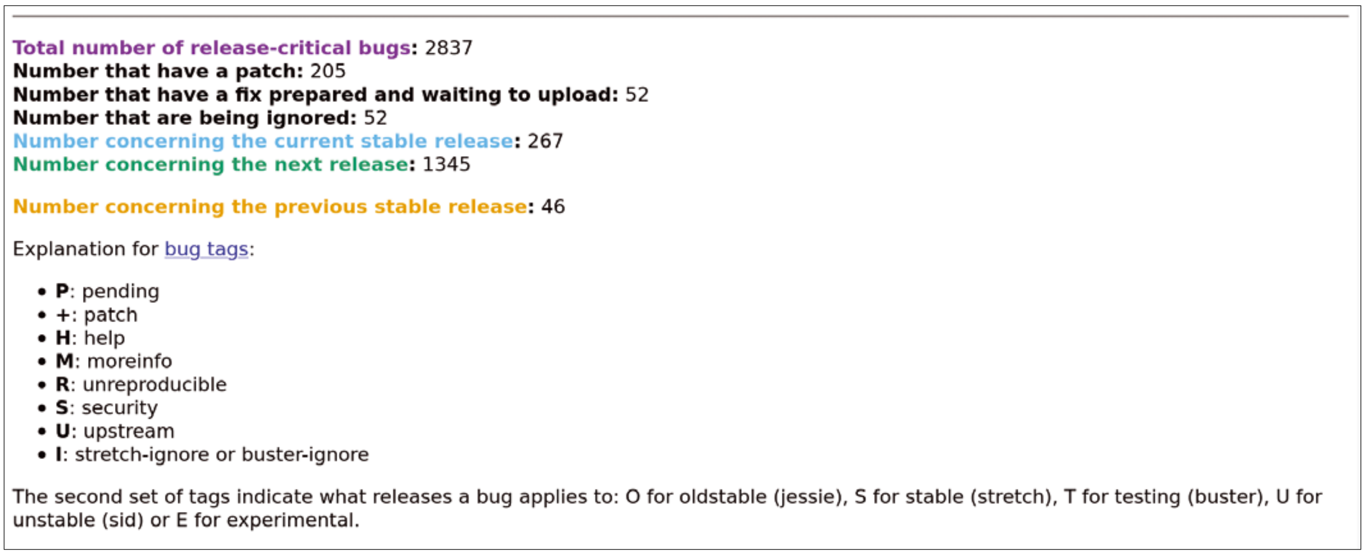
```
┤ Select a file (installed:krita) ├

/usr/share/doc/krita/changelog.Debian.amd64.gzchangelog.Debian.amd64.gz
/usr/share/doc/krita/changelog.Debian.gz       changelog.Debian.gz
/usr/share/doc/krita/copyright                 copyright


            <Ok>                              <Cancel>
```

Figure 6: With `debmany`, you can generate a list of all the man pages associated with a package and then select the ones to view.



```
~$ check-enhancements gimp
mp-gmic:          Installed: (none)        Candidate: 2.4.5-1
mp-gutenprint:            Installed: (none)        Candidate: 5.3.1-7
mp-help-ca:       Installed: (none)        Candidate: 2.8.2-1
mp-help-de:       Installed: (none)        Candidate: 2.8.2-1
mp-help-el:       Installed: (none)        Candidate: 2.8.2-1
```

Figure 7: Using `check-enhancements` lists packages that add functionality to a package but are not required to run it.



Total number of release-critical bugs: 2837
Number that have a patch: 205
Number that have a fix prepared and waiting to upload: 52
Number that are being ignored: 52
Number concerning the current stable release: 267
Number concerning the next release: 1345

Number concerning the previous stable release: 46

Explanation for bug tags:

- **P**: pending
- **+**: patch
- **H**: help
- **M**: moreinfo
- **R**: unreproducible
- **S**: security
- **U**: upstream
- **I**: stretch-ignore or buster-ignore

The second set of tags indicate what releases a bug applies to: O for oldstable (jessie), S for stable (stretch), T for testing (buster), U for unstable (sid) or E for experimental.

Figure 8: With `popbugs`, you can correlate the most often used packages on your system with critical bugs.

tracking down which related products might possibly be the problem. No options are available – just a history of past problems. This information may not be much use if the package whose installation attempt started the problem is related to dozens of others. Note, too, that a general release upgrade is listed as breaking packages (Figure 9).

checkrestart

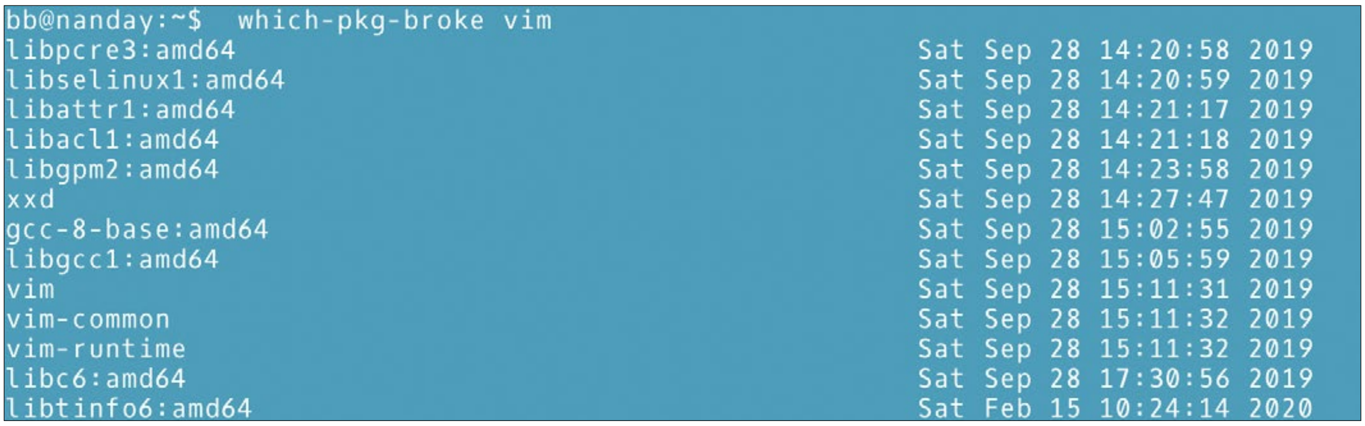
Run by root, `checkrestart` (Figure 10) finds services that need to be restarted after a system upgrade that has added updated or new libraries, because the processes running still expect the old deleted versions. It can be run with `--package` (`-p`), so that only deleted package files are located, or with `--all` (`-a`), which may produce false positives such as a file in the `/tmp` directory. In addition, results can be edited using `--blacklist=FILE` (`-b=FILE`) to create a list of regular expressions to be excluded from the results. Still other ways to limit results are to exclude files associated with listed services with `--ignore=SERVICE` (`-i=NAME`) or process IDs with `--excludepid=PID` (`-e PID`). Output length can be controlled either with the usual `--verbose` option or with `--terse` (`-t`).

A Package Toolkit

Perhaps the reason that Debian Goodies is relatively unknown is that it is only about 15 years old. All the same, if you work with packages in any capacity, Debian Goodies' scripts are worth getting to know. Should your Debian-derivative distribution not carry the package, you can probably use Debian's own package. These small scripts can be invaluable for all maintenance connected to packages. If you are troubleshooting, they can save you hours of work. ■■■

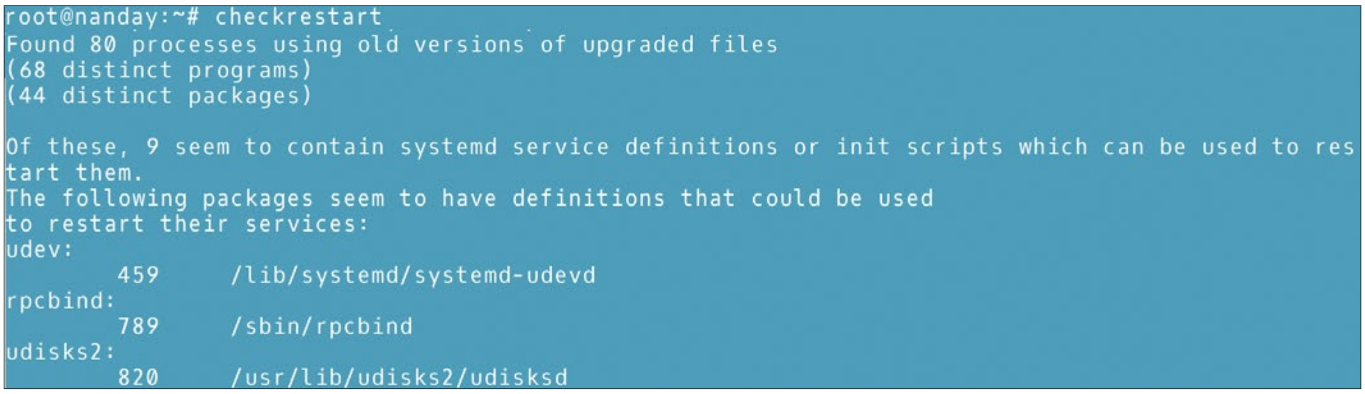
Info

[1] Debian Goodies: *https://packages.debian.org/search?keywords=debian+goodies&searchon=names&suite=stable&section=all*



```
bb@nanday:~$  which-pkg-broke vim
libpcre3:amd64                                        Sat Sep 28 14:20:58 2019
libselinux1:amd64                                     Sat Sep 28 14:20:59 2019
libattr1:amd64                                        Sat Sep 28 14:21:17 2019
libacl1:amd64                                         Sat Sep 28 14:21:18 2019
libgpm2:amd64                                         Sat Sep 28 14:23:58 2019
xxd                                                   Sat Sep 28 14:27:47 2019
gcc-8-base:amd64                                      Sat Sep 28 15:02:55 2019
libgcc1:amd64                                         Sat Sep 28 15:05:59 2019
vim                                                   Sat Sep 28 15:11:31 2019
vim-common                                            Sat Sep 28 15:11:32 2019
vim-runtime                                           Sat Sep 28 15:11:32 2019
libc6:amd64                                           Sat Sep 28 17:30:56 2019
libtinfo6:amd64                                       Sat Feb 15 10:24:14 2020
```

Figure 9: **When using `which-pkg-broke`, remember that a general release is listed as breaking packages, even though there was no difficulty.**



```
root@nanday:~# checkrestart
Found 80 processes using old versions of upgraded files
(68 distinct programs)
(44 distinct packages)

Of these, 9 seem to contain systemd service definitions or init scripts which can be used to res
tart them.
The following packages seem to have definitions that could be used
to restart their services:
udev:
        459     /lib/systemd/systemd-udevd
rpcbind:
        789     /sbin/rpcbind
udisks2:
        820     /usr/lib/udisks2/udisksd
```

Figure 10: **After a system upgrade, `checkrestart` shows which services need to be restarted.**

The sys admin's daily grind: DenyHosts

Smart Doorkeeper

When it comes to warding off unwanted login tests on SSH port 22 and others, Charly likes to keep an ace or two up his sleeve by relying on DenyHosts instead of Fail2ban. *By Charly Kühnast*

Many sys admins use Fail2ban to keep unwanted login tests on SSH port 22 and others at bay. However, there are alternatives to Fail2ban. Today, I would like to introduce you to DenyHosts, which does basically the same thing as Fail2ban but has one or two aces up its sleeve.

You can find DenyHosts as part of the standard package on all popular distributions; my lab system is an up-to-date Debian. To install DenyHosts, type:

```
sudo apt install denyhosts
```

This is followed by the most important step of the configuration: You need to add the IP addresses of all the systems from which you want to log in to the protected system without being blocked to the `/etc/hosts.allow` file (Listing 1). If you forget this step, there is a danger that you will be locked out of the system permanently – so it might be better to take another sip of coffee and check again that there are no typos.

DenyHosts is controlled by the `/etc/denyhosts.conf` file, which contains sensible defaults, but I'll take a closer look at some of them anyway. The most important lines in this longish configuration file are the ones where I define the number of failed attempts before the system locks out a host by applying an iptables rule. In doing so, DenyHosts takes into account whether you log in with a username that actually exists in `/etc/passwd` (Listing 2). After adjusting the values to your satisfaction, it's time to put DenyHosts into operation (Listing 3).

DenyHosts writes all activities to the `/var/log/auth.log` file (Figure 1). If you exceed one of the thresholds, the IP address of the knocking system is entered in the `/etc/hosts.deny` file and blocked by iptables (Figure 2). As a highlight, DenyHosts can synchronize with other servers worldwide – the settings for this can be found a bit further down in the `Sync` section of the configuration file.

The fact that DenyHosts, unlike Fail2ban, does not release a blocked IP address after a certain time is seen as a disadvantage by some admins. So far, however, none of my users has managed to enter their password incorrectly more than 10 times. In the end, it depends on your personal preferences as to which of the two tools you use. I tend cautiously towards DenyHosts. ■■■

Listing 1: hosts.allow

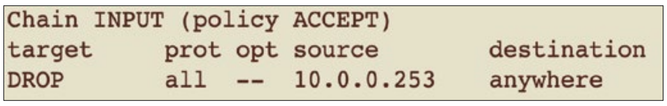
```
sshd: 10.0.0.8
sshd: 10.0.0.42
[...]
```

Listing 2: denyhosts.conf

```
# Admissible failed attempts:
# Username does not exist
DENY_THRESHOLD_INVALID = 5
# Username exists
DENY_THRESHOLD_VALID = 10
# Username is "root"
DENY_THRESHOLD_ROOT = 1
```

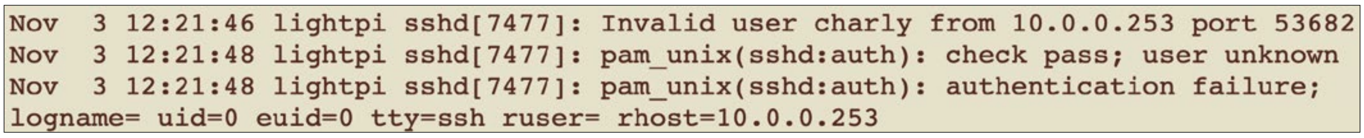
Listing 3: Starting DenyHosts

```
$ sudo systemctl restart denyhosts
$ sudo systemctl enable denyhosts
```



```
Chain INPUT (policy ACCEPT)
target     prot opt source         destination
DROP       all  --  10.0.0.253     anywhere
```

Figure 2: Game over: Iptables has blocked the IP address.



```
Nov  3 12:21:46 lightpi sshd[7477]: Invalid user charly from 10.0.0.253 port 53682
Nov  3 12:21:48 lightpi sshd[7477]: pam_unix(sshd:auth): check pass; user unknown
Nov  3 12:21:48 lightpi sshd[7477]: pam_unix(sshd:auth): authentication failure;
logname= uid=0 euid=0 tty=ssh ruser= rhost=10.0.0.253
```

Figure 1: DenyHosts logs its actions in the `/var/log/auth.log` file.

Author

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

Exploring the RISC-V processor architecture

New Chip in Town

The open source RISC-V processor architecture is poised to shake up the processor industry. Thanks to the Qemu emulator, you can get to know the RISC-V without waiting for affordable hardware.

By Eva-Katharina Kunst and Jürgen Quade

Most of the IT industry has consolidated around Intel-equivalent processors with the x86 and (related) AMD64 architectures. But just two decades ago, several other processor manufacturers and architectures were fighting for a place in the market. Many of these processors (including MIPS, PowerPC, Alpha, and SPARC) followed a set of principles for processor design known as Reduced Instruction Set Computer (RISC). RISC systems live on today, most notably in the ARM architecture used in today's in smartphones, Raspberry Pis, and other electronic devices. (The acronym ARM actually stands for *Advanced RISC Machine*.) But another RISC contender is also on the rise.

RISC-V, pronounced "Risk Five," is widely hailed as a shooting star in the processor architecture sky [1]. Proponents believe RISC-V can do everything ARM can do – and it is all open source and royalty free. The non-proprietary nature of RISC-V means it could allow small chip makers to get in a game that is now dominated by a few tech giants. And even mainstream vendors

Hardware Options

The ESP32-C3-DevKitC-02 board, which features a RISC-V core, is available in mail order shops for prices starting at around EUR10 (Figure 1). Chinese manufacturer Espressif has added a RISC-V-based ESP32-C3 to its well-known and successful ESP32 model; it has slightly slower performance than the classic model, but it is smaller and less expensive.

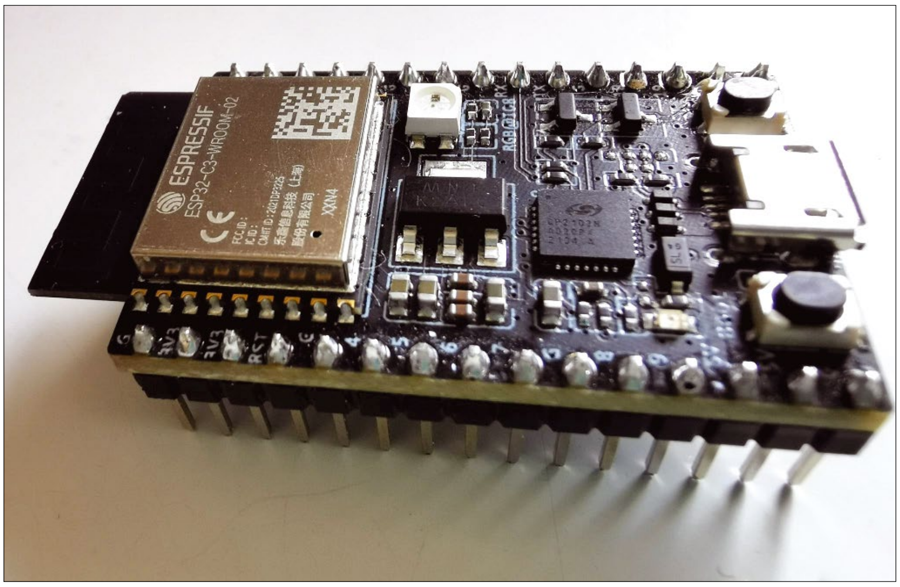


Figure 1: Espressif is already offering its ESP32-C3-DevKitC-02 RISC-V board for a low price.

Photo by Jason Jarrach on Unsplash

Figure 2: The RISC-V logo.

Table 1: RISC-V Features

Modularity	Basic Instruction Set Plus Extensions
Register width	32, 64, 128
No. of registers	16 (for embedded), 32, 64
Architecture type	Load/Store
File storage format	Little Endian
Number of instructions	40 (RV32I, Computational, Control Flow, Memory Access)
Privilege Levels	3
Machine type	Three register machine
Instruction types	6 (R, I, U, S, B, J)
Software	Comprehensive software ecosystem with popular software stacks (Linux, FreeRTOS)
Misc.	Stable specification, no further changes planned

could one day gain an edge by choosing RISC-V and avoiding licensing fees.

At just 10 years old, the RISC-V architecture is still considered comparatively young. If you can live without Linux and prefer FreeRTOS, you can get started today and gain some RISC-V experience on reasonably priced hardware (see the box entitled "Hardware Options"). Until recently, if you were looking for a RISC-V processor capable of running Linux, you would have to pay as much as EUR600. As this issue goes to print, an affordable, Linux-ready RISC-V board has finally reached the market, though little is known about it [2].

If you want to get to know RISC-V in combination with Linux, you can do so right now without waiting for the perfect hardware. A RISC-V compiler is available for Ubuntu, and the Qemu hardware emulator is also available to emulate a complete 32- or 64-bit RISC-V system. Some Linux distros already have RISC-V variants of their software collections, so you can quickly get to work.

This article shows how to take your first steps with exploring Linux on RISC-V – but first a little background.

About RISC-V

Ten years ago, computer scientists at the University of California, Berkeley (UC Berkeley), developed a new CPU instruction set architecture (ISA) known as RISC-V (Figure 2); the idea was for RISC-V to be modern and suitable for both teaching and research, as well as actual applications [3]. The special highlight was the BSD-based licensing model, which allows the licensee to use the architecture free of charge (Table 1).

Over the past few years, companies large and small have begun to explore the possibilities of RISC-V; developing, producing, and distributing hardware with a RISC-V core; and making their designs available free of charge. The large Chinese online retailer Alibaba, for example, has completely disclosed the RISC-V core it developed, as has disk manufacturer Western Digital. Even Apple, which has a very powerful and efficient ARM CPU up its sleeve in the form of its M1 chip, has recently been looking for RISC-V developers.

Much like ARM, but in total contrast to x86, the RISC-V architecture scales very well to small sizes. In addition to variants that have a full-blown memory management unit (MMU) and are therefore suitable for Linux, RISC-V supports very simple microcontrollers for bare-metal programming, which makes it suitable for programming that does not require system software or is equipped with the FreeRTOS real-time operating system.

The simplest RISC-V variants implement only the core instructions, also known as the I-instructions. In addition to logical operations, these core instructions consist of just adding and subtracting; multiplying and dividing are reserved for the M-extension. Table 2 lists examples of RISC-V extensions, such as the F extension for floating point or the C extension for compressed instructions. UC Berkeley suggests implementing at least the I, M, F, D, and A extensions and uses the letter G (for "general") to abbreviate this architecture to RV32G.

RISC-V is defined as a 32-, 64-, and even 128-bit architecture. In other words: All internal registers have a width of 32, 64, or even 128 bits, depending on the architecture. There are no less than 32 or 64 of these registers, which are numbered *X0* to *X31* or *X63*. Apart from the register *X0*, which can only be read and where all bits are set to 0, the registers can be used universally. For example, there is no special stack pointer. However, the architecture provides for a separate program counter (PC) (i.e., an instruction counter that contains the address of the instruction that the CPU needs to process next).

Whereas the x86 architecture is a two-register machine, RISC-V typically has three registers involved in an operation. If a two-register machine adds the contents of two registers, it has to store the results in one of the two registers. A three-register machine, on the other hand, can store the result in a third register (*X1* = *X2* + *X3*).

Table 2: A Selection of RISC-V Instruction Set Extensions

RV32I	Base Instruction Set
RV32A	A extension (atomic read-modify-write)
RV32B	B extension (bit manipulation)
RV32C	C extension (compressed instructions)
RV32D	D extension (double precision floating point)
RV32F	F extension (single precision floating point)
RV32M	M extension (multiplication, division)
RV32S	S extension (supervisor level instructions)
RV32V	V extension (vector instructions)

Listing 1: RISC-V Ubuntu for Qemu

```
#!/bin/bash
BASISDIR=~/risc-v
UBUNTU_IMG=ubuntu-20.04.3-preinstalled-server-riscv64+unmatched.img

if test ! -f `type -p riscv64-linux-gnu-gcc`; then
  echo "install compiler..."
  sudo apt install gcc-riscv64-linux-gnu
fi
if test ! -f `type -p qemu-system-riscv64`; then
  echo "install qemu..."
  sudo apt install qemu-system-misc qemu-utils
fi
if test ! -d ${BASISDIR}; then
  echo "mkdir ${BASISDIR}"
  mkdir ${BASISDIR}
fi
cd ${BASISDIR}

if test ! -f opensbi/build/platform/generic/firmware/fw_payload.bin; then
  wget https://cdimage.ubuntu.com/releases/20.04/release/${UBUNTU_IMG}.xz

  xz -dk ${UBUNTU_IMG}.xz
  qemu-img resize -f raw ${UBUNTU_IMG} +6G
  export CROSS_COMPILE=riscv64-linux-gnu-
  git clone https://source.denx.de/u-boot/u-boot.git
  cd u-boot/
  git reset --hard v2021.10-rc3
  make qemu-riscv64_smode_defconfig
  make -j$(nproc)
  cd ..
  git clone https://github.com/riscv/opensbi.git
  cd opensbi/
  make PLATFORM=generic FW_PAYLOAD_PATH=../u-boot/u-boot.bin
  cd ..
fi

qemu-system-riscv64 -machine virt -m 4G -nographic \
  -bios opensbi/build/platform/generic/firmware/fw_payload.bin \
  -smp cores=2 -gdb tcp::1234 \
  -device virtio-net-device,netdev=eth0 \
  -netdev user,id=eth0,hostfwd=tcp::2222-:22 \
  -drive if=none,file=${UBUNTU_IMG},format=raw,id=mydisk \
  -device ich9-ahci,id=ahci -device ide-hd,drive=mydisk,bus=ahci.0 \
  -device virtio-rng-pci
```

Installing RISC-V Linux

You can't just call `apt install` on Ubuntu to get started. In fact, for an emulated RISC-V machine, you need a BIOS, a bootloader, and finally, the Linux operating system itself.

This all starts with the RISC-V compiler, which you install by typing

```
apt install gcc-riscv64-linux-gnu
```

Download the latest version of the OpenSBI open source BIOS and generate the BIOS. The bootloader is U-Boot, which is well known in the embedded area – you will need to compile a RISC-V version of U-Boot.

The RISC-V server image [4] hosted by Canonical targets a small memory footprint at 466MB. You still need to convert this to a usable size after downloading and unpacking the image using the Qemu tools. Whether this size will be 20GB or just 5GB is something you'll need to decide for yourself.

Finally, you need to launch Qemu with a plethora of parameters, especially for the network connection, if you want to use Internet access. User *Aaronfranke* provides a script [5] that maps the essential installation steps. Listing 1 shows a variant that we modified and extended; the script optimizes, adjusts, and conjures up a bootable system more or less automatically.

The script creates a new `risc-v/` subfolder in your home directory, installs the necessary software by running `apt install`, and reloads the other required components, including the pre-built Ubuntu image off the web. It generates the firmware and the bootloader and finally starts RISC-V Linux. Incidentally, the download and installation only occur on the first call.

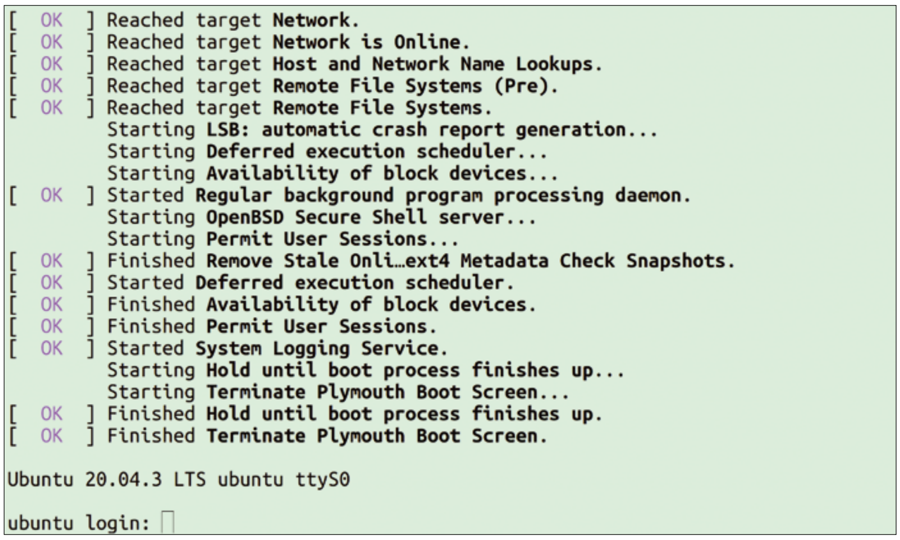
Save the script in a file named, say, `risc-v-linux.sh`, change the access rights to execute by typing

```
chmod 755 risc-v-linux.sh
```

and finally start the installation. Launch Linux in the console by calling

```
./risc-v-linux.sh
```

At this point, however, you will need some patience, because the RISC-V CPU implementation in Qemu is a software emulation rather than virtualization. A few minutes later, however, you should



```
[  OK  ] Reached target Network.
[  OK  ] Reached target Network is Online.
[  OK  ] Reached target Host and Network Name Lookups.
[  OK  ] Reached target Remote File Systems (Pre).
[  OK  ] Reached target Remote File Systems.
         Starting LSB: automatic crash report generation...
         Starting Deferred execution scheduler...
         Starting Availability of block devices...
[  OK  ] Started Regular background program processing daemon.
         Starting OpenBSD Secure Shell server...
         Starting Permit User Sessions...
[  OK  ] Finished Remove Stale Onli…ext4 Metadata Check Snapshots.
[  OK  ] Started Deferred execution scheduler.
[  OK  ] Finished Availability of block devices.
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started System Logging Service.
         Starting Hold until boot process finishes up...
         Starting Terminate Plymouth Boot Screen...
[  OK  ] Finished Hold until boot process finishes up.
[  OK  ] Finished Terminate Plymouth Boot Screen.

Ubuntu 20.04.3 LTS ubuntu ttyS0

ubuntu login: 
```

Figure 3: The login screen for RISC-V Ubuntu.

be prompted to log in, and you can comply by entering the username *ubuntu* with the initial password of *ubuntu* (Figure 3). The system will then force you to change the password for security reasons.

No Command-Line Fun

Since the serial console has some weaknesses, it is best to log in to the new system via SSH using another terminal window. Qemu configured our script for SSH login, so you can type

```
ssh -p 2222 ubuntu@localhost
```

Now enter the `uname -a` command, and you will see that you are working on a RISC-V machine. `cat /proc/cpuinfo` also shows the architecture, including the implemented extensions, but is otherwise surprisingly terse. `cat /proc/interrupts`, `dmesg`, or `file /usr/bin/bash` show further traces of the RISC-V architecture (Figure 4). On the whole, however, Linux lives up to its reputation as a platform-independent operating system by carefully concealing the new CPU architecture under the hood.

Use `apt update` to update the package list in the usual way. But avoid `apt upgrade`, which could swap the kernel and bootloader and cause problems. Generally speaking, the software will not be one hundred percent stable across the board. Once you have installed a compiler and developer tools by typing

```
apt install build-essential gdb vim
```

you can enter and compile a small Hello RISC-V program. Then check out the CPU registers and assembler code in the GNU debugger (Figure 5).

By the way, the CPU registers do not use the previously mentioned register designations *X0* to *X31* in the debugger or in the output of the `dmesg --kernel` command. Instead, you will see the names `ra`, `sp`, `t0`, `s6`, and so on. The RISC-V Foundation proposed – or specified – a convention for the use of the registers, and the programmers or the manufacturers of the development tools also adhere to this convention. According to the convention, for example, register *X2* acts as a stack pointer (`sp`); *X1* contains the return address (`ra`) for a subroutine call. Registers with a *T* in their names are used to cache data, and those that start with an *A* contain the call parameters when a function is called.

Developers don't have to worry about these details in most cases; they are mainly relevant for kernel programmers when debugging.

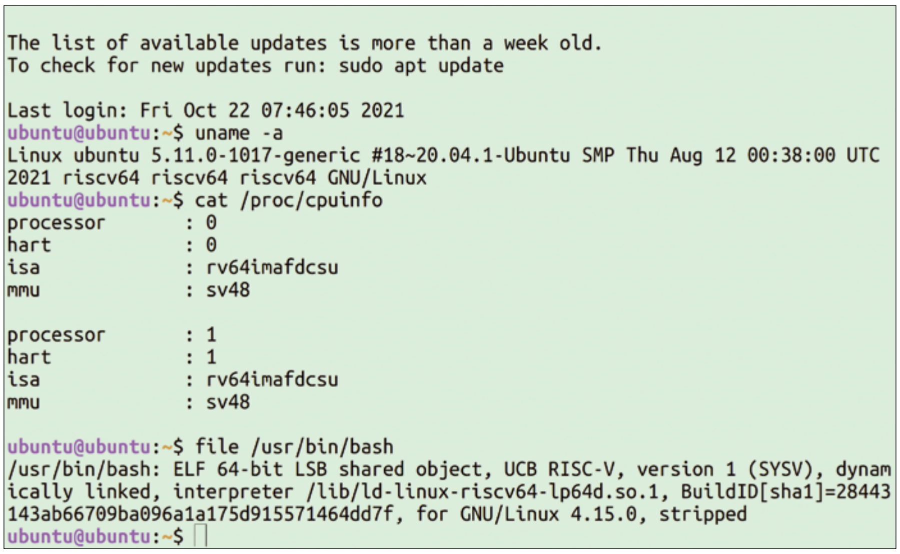
Quitting Qemu

At the end of the day, shut down RISC-V Ubuntu cleanly in the terminal by typing `sudo poweroff`. To then exit Qemu, use the keyboard shortcut Ctrl+A,X.

Perfect Match

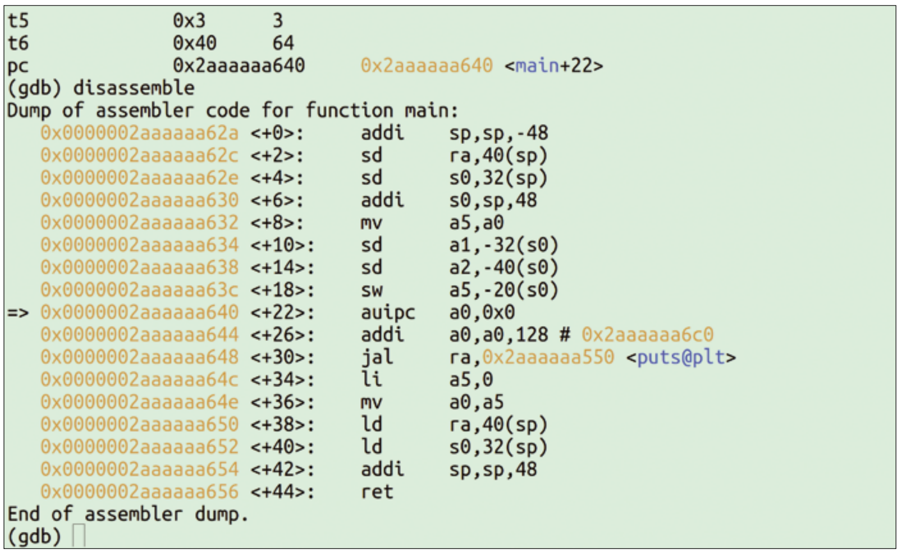
In addition to Ubuntu, Fedora [6] and Debian [7] also provide prebuilt RISC-V images. The Debian variant we tested, however, did act up a little and did not let us install our favorite editor Vim. Anyway, it's more exciting to use a DIY build instead of a pre-built system image. You can use the Buildroot system builder for this; you'll find a short tutorial online [8].

As you can see, Linux users have no reason to complain about a lack of support for the RISC-V instruction set architecture. This should not come as a surprise: With a smart, modular, open-source design, plus a free license – Linux and RISC-V are a perfect match. ■■■



```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Fri Oct 22 07:46:05 2021
ubuntu@ubuntu:~$ uname -a
Linux ubuntu 5.11.0-1017-generic #18~20.04.1-Ubuntu SMP Thu Aug 12 00:38:00 UTC
2021 riscv64 riscv64 riscv64 GNU/Linux
ubuntu@ubuntu:~$ cat /proc/cpuinfo
processor       : 0
hart            : 0
isa             : rv64imafdcsu
mmu             : sv48

processor       : 1
hart            : 1
isa             : rv64imafdcsu
mmu             : sv48

ubuntu@ubuntu:~$ file /usr/bin/bash
/usr/bin/bash: ELF 64-bit LSB shared object, UCB RISC-V, version 1 (SYSV), dynam
ically linked, interpreter /lib/ld-linux-riscv64-lp64d.so.1, BuildID[sha1]=28443
143ab66709ba096a1a175d915571464dd7f, for GNU/Linux 4.15.0, stripped
ubuntu@ubuntu:~$ 
```

Figure 4: Traces of the RISC-V architecture are easy to find.



```
t5             0x3      3
t6             0x40     64
pc             0x2aaaaaa640      0x2aaaaaa640 <main+22>
(gdb) disassemble
Dump of assembler code for function main:
   0x0000002aaaaaa62a <+0>:     addi    sp,sp,-48
   0x0000002aaaaaa62c <+2>:     sd      ra,40(sp)
   0x0000002aaaaaa62e <+4>:     sd      s0,32(sp)
   0x0000002aaaaaa630 <+6>:     addi    s0,sp,48
   0x0000002aaaaaa632 <+8>:     mv      a5,a0
   0x0000002aaaaaa634 <+10>:    sd      a1,-32(s0)
   0x0000002aaaaaa638 <+14>:    sd      a2,-40(s0)
   0x0000002aaaaaa63c <+18>:    sw      a5,-20(s0)
=> 0x0000002aaaaaa640 <+22>:    auipc   a0,0x0
   0x0000002aaaaaa644 <+26>:    addi    a0,a0,128 # 0x2aaaaaa6c0
   0x0000002aaaaaa648 <+30>:    jal     ra,0x2aaaaaa550 <puts@plt>
   0x0000002aaaaaa64c <+34>:    li      a5,0
   0x0000002aaaaaa64e <+36>:    mv      a0,a5
   0x0000002aaaaaa650 <+38>:    ld      ra,40(sp)
   0x0000002aaaaaa652 <+40>:    ld      s0,32(sp)
   0x0000002aaaaaa654 <+42>:    addi    sp,sp,48
   0x0000002aaaaaa656 <+44>:    ret
End of assembler dump.
(gdb) 
```

Figure 5: Disassembling a small "Hello, RISC-V" program in the GNU debugger.

Info

[1] RISC-V specifications: *https://riscv.org/technical/specifications/*

[2] New Linux-ready RISC-V board: *https://www.aliexpress.com/item/1005003594875290.html*

[3] RISC-V: *https://en.wikipedia.org/wiki/RISC-V*

[4] RISC-V on Ubuntu: *https://wiki.ubuntu.com/RISC-V*

[5] "Ubuntu Server on RISC-V documentation needs updating": *https://discourse.ubuntu.com/t/ubuntu-server-on-risc-v-documentation-needs-updating/23927/10*

[6] RISC-V on Fedora: *https://fedoraproject.org/wiki/Architectures/RISC-V*

[7] RISC-V on Debian: *https://wiki.debian.org/RISC-V*

[8] Qemu and RISC-V: *https://wiki.amarulasolutions.com/bsp/riscv/qemu.html*

Authors

Eva-Katharina Kunst has been a fan of open source since the early days of Linux. **Jürgen Quade**, a professor at the Niederrhein University of Applied Sciences, also conducts training courses for companies on the subjects of driver programming and embedded Linux.

Host-INT brings network packet telemetry to Linux hosts

Packet Timer

Inband Network Telemetry and Host-INT can provide valuable insights on network performance – including information on latency and packet drops. *By Andrew Fingerhut and Deepa Seshadri*

Hyperscale data centers are seeking more visibility into network performance for better manageability. This challenge is made more difficult as the number of switches and servers grow, and as data flows evolve to 100Gbps and higher. Knowing where network congestion is and how it is affecting data flows and service level agreements (SLAs) is critical.

Inband Network Telemetry (INT) is an open specification from the P4 open source community [1]. The goal of INT is to allow the collection and reporting of packets as they flow through the network. Intel has brought this capability to the Linux open source community with Host Inband Network Telemetry (Host-INT). The Host-INT framework can collect some very valuable data on network conditions – such as latency and packet drops – that would be very hard to obtain otherwise. Host-INT is ideal for app developers who need to know the network's impact on an application. Or, a large wide area network (WAN) service provider could use Host-INT to ensure that its service level agreements are being met.

Host-INT builds on Switch-INT, another P4 implementation that performs measurements on network packets. Both Host-INT and Switch-INT are designed to operate entirely in the data plane of a network device (e.g., in the hardware fast path of a switch ASIC or Ethernet network adapter). Switch-INT is currently running on programmable switch infrastructures such as Intel's Tofino Intelligent Fabric Processors [2]. By operating in the data plane, Switch-INT can provide extensive network telemetry without impacting performance.

How Host-INT Works

Host-INT is implemented in the Linux server, not in the switches, and it thus looks at packet flows from outside the network. The INT source node, located at the network ingress, inserts instructions and network telemetry information into the packet. The INT sink, located at the network egress, removes the INT header and metadata from the packet and sends it to the telemetry collector.

The Host-INT configuration consists of a source host, where packets are modified to include the telemetry information, and the destination host, where metadata is collected and reported to an analytics program (Figure 1). The result is that Host-INT is able to collect difficult-to-obtain performance data. A DevOps team, for example, could use Host-INT to study the impact of the network on their app's performance – and obtain much more detailed information than would be available using conventional tools like `ping` and `traceroute`.

Also, organizations that depend on large WANs can use the latency and packet drop data that comes from Host-INT to ensure service levels and connect early with their service provider to minimize repair time.

Host-INT Collects Metadata

The Host-INT specification describes several options for collecting additional metadata from packet headers in order to measure latency, depth of the queues that the packet passes through, and link utilization levels.

After installing and enabling Host-INT on Linux servers (check the Host-INT GitHub page [3] for more on supported Linux distributions and versions), INT headers are added to IPv4 TCP and UDP packets that are then read by another INT-enabled host on the network. For example, if you configure Host-INT on host A to add INT headers to all packets sent from host A to host B, an extended Berkeley Packet Filter (eBPF) program loaded into the Linux kernel on host A modifies all TCP and UDP packets destined for host B.

The INT header contains:

- The time when the packet was processed.
- A sequence number that starts at 1 for the first packet of each flow and increments for each subsequent packet of the flow.

Host-INT's notion of a flow is all packets with the same 5-tuple (i.e., the same combination of values for these five packet header fields):

- IPv4 source and destination address
- IPv4 protocol
- TCP/UDP source and destination port

The packet encapsulation format used in Host-INT's initial release is to add the INT headers after the packet's TCP or UDP header. In order for the receiving host B to distinguish packets that have INT headers from those that do not, host A also modifies the value of the differentiated services field codepoints (DSCP) within the IPv4 Type of Service field to a configurable value. In practice it is good to consult with a network administrator to find a DSCP value that is not in use for other purposes within the network.

When packets arrive at host B, another eBPF program running in host B's kernel processes it. If the packet has an INT header (determined by the DSCP value), the eBPF program removes it and sends the packet on to the Linux kernel networking code, where it will then proceed to the target application.

Before removing the INT header, host B's eBPF program calculates the one-way latency of the packet, calculated as the time at host B when the packet was received, minus the timestamp that host A sent in the INT header. Note that any inaccuracy in the synchronization of the clocks on host A and host B introduces an error into this one-way latency measurement. It is recommended that you use Network Time Protocol (NTP) or Precision Time Protocol (PTP) to synchronize the clocks for devices on your network.

Host B's eBPF program looks up the packet's flow (determined by the same 5-tuple of packet header fields used at the source host A) in an eBPF map. Independently, for the packets that host B receives with INT headers, it keeps the following data:

- The last time a packet was received.
- The one-way latency of the previous packet received by host B for this flow.
- A small data structure that, used in combination with the sequence number in the INT header, determines how many packets were lost in the network for this flow.

Host B will generate an INT report packet upon any of these events:

- The first packet of a new flow is received with an INT header.
- Once-per-packet drop report time period. Every flow is checked to see if any new packet drops have been detected since the previous period. If there have been new packet losses detected recently, an INT loss report packet is generated. The report contains the number of lost packets detected and the header of the packet.
- If the one-way latency of the current packet is significantly different from the latency of the previous packet for this flow, an INT latency report packet is generated.

Thus, by looking at the stream of INT report packets, Host-INT can monitor the following things:

- Every time the latency of a flow changes significantly
- Periodic updates of the lost packet count

Both of these things are reported on a per-flow basis.

Host-INT generates INT report packets that are sent from INT packet receivers back to the sending host, where the data extracted from the report is written to a text file. Adding other components expands the functionality of Host-INT. Host-INT supports extensions that send INT report packets to the host that sent the packet containing the header – and even to a pre-configured IP address.

The typical reason to use a pre-configured IP address is to send INT reports from many hosts to a telemetry analytics system that can collect, collate, analyze, and display telemetry data so operators can make insightful decisions about their network. (Intel's Deep Insight Analytics software [4] is an example of an analytics system that can process INT reports.)

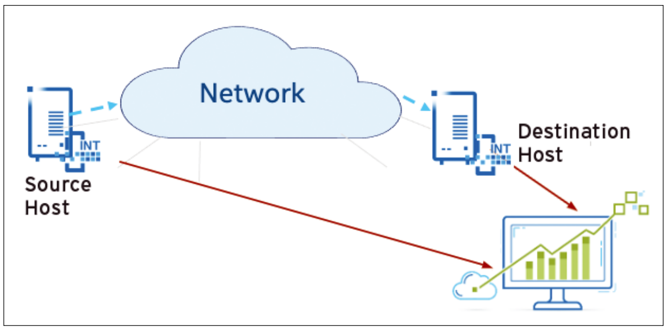


Figure 1: **Host-INT in action: The source host adds an INT header to the packet. The destination host extracts the header and reports telemetry metadata to an analytics program.**

Current Limitations

Although Host-INT's reports can help network administrators learn about flows experiencing high latency or many packet drops, Host-INT cannot currently help narrow down the root cause within a network for this behavior. Switch-INT, however, does have the ability to help find the root cause. If you deploy INT-capable switches on your network, you can configure those switches to generate INT reports on events such as packet drops or packet queues causing high latency. By configuring your switches properly, you can quickly determine the actual location in the network where congestion and packet drops occur. You can also determine snapshots of other packets that were in the same queue as the dropped or high-latency packets.

Conclusion

Host-INT adds packet telemetry to Linux hosts, providing an effective way to monitor network conditions in data centers, including congestion, packet drops, and latency. Host-INT works to provide an outside-in look at network performance, and you can combine it with Switch-INT for a more comprehensive management tool. The software and more documentation are available on GitHub [3]. ■■■

Info

[1] P4 open source programming language: *https://p4.org/*

[2] Tofino Intelligent Fabric Processors: *https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html*

[3] Host-INT on GitHub: *https://github.com/intel/host-int*

[4] Deep Insight Analytics: *https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/network-analytics/deep-insight.html*

Author

Andrew Fingerhut is a Principal Engineer at Intel Corporation working on programmable switches and NICs, and has nearly 30 years of experience in computer networking. He also actively participates in the P4 programming language through P4.org working groups.

Author

Deepa Seshadri is a Product Manager for the Intel Deep Insight Network Analytics software at Intel Corporation. Her area of expertise lies in productizing new software technology with concept, all the way to production.

Getting Ready for PipeWire

Heir Apparent

In the coming year, PipeWire will replace PulseAudio, resulting in better audio on Linux. If you can't wait, here's what you need to know to get started with PipeWire. *By Bruce Byfield*

Unless you use a version of Fedora released in 2021, you may not have heard of PipeWire [1]. However, by this time next year, PipeWire will likely be installed on your computer. Already, many distributions are starting to carry PipeWire (marked as experimental) in their repositories. Still unfinished and its installation varying, depending on distribution, PipeWire is about to replace PulseAudio as Linux's main audio server. If you are unwilling to wait until PipeWire becomes a standard part of a Linux installation, here is what you should know.

PipeWire was created by Wim Taymans of Red Hat in 2015. Based on an earlier project called PulseVideo, PipeWire was originally intended as a server for capture and playback of audio and video. The video side of the project is still in development, but the audio side is mature enough that in the spring of 2021 Fedora 34 become the first Linux distribution to install it by default. In Fedora 34, PipeWire is used to manage PulseAudio, JACK, ALSA, and GStreamer-based applications.

Given that Red Hat developed PulseAudio earlier as an audio manager, the reasoning behind rushing PipeWire into general use is being silently passed over. However, the rush is almost certainly due to numerous complaints about PulseAudio. Despite being the most common audio server on Linux, PulseAudio has become infamous for its awkward interface and, especially in its early days, for the project's slowness to respond to numerous bug reports. A recent Google search on "problems with Pulse Audio" [2] returned 289,000 results. At the top of these results is a list of common issues:

- Restarting the PulseAudio daemon
- Interrupting play in Amarok when running Skype
- Sound level is low or suddenly becomes too loud
- Missing playback devices or audio capture
- Front panel jacks not working
- Stuttering and audio interruptions

Photo by Nathan Mcgregor on Unsplash

- Excessive CPU usage and distortion
- Various problems with Skype and Wine

Just below this list in the results is a Reddit thread asking, “Why do people dislike PulseAudio?” [3] The thread has 94 comments, many of which list additional problems and, at times, voice considerable frustration. For such a prominent piece of Linux desktop structure, PulseAudio has been an embarrassment that has never entirely gone away.

None of this dissatisfaction, of course, is officially mentioned. Instead, the emphasis is on the very real technical improvements offered by PipeWire. The project’s home page explains, “PipeWire was designed with a powerful security model that makes interacting with audio and video devices from containerized applications easy, with supporting Flatpak applications being the primary goal. Alongside Wayland and Flatpak, we expect PipeWire to provide a core building block for the future of Linux application development.” In the project’s GitLab repository, a section entitled “How Is PipeWire Supposed to Be a Better PulseAudio?” [4] includes the following advantages:

- Lower latency with less CPU usage and fewer dropouts
- A security model with greater privacy
- More control over how applications are linked to devices and filters
- An external policy manager that allows greater integration into a desktop system

A similar heading, “How Is PipeWire Supposed to Be a Better JACK?”[5], lists the following advantages over JACK:

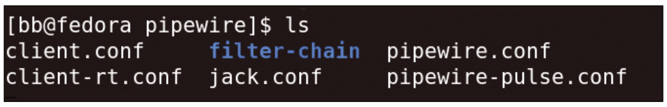
- Greater efficiency for merging devices and resampling
- Multiple devices only need to be resampled together when linked to each other
- Better interaction with Bluetooth and any similar future devices
- Dynamic adaption of latency, which improves power usage on laptops
- The ability to implement exclusive access to devices, potentially improving audio quality and power usage.
- Full latency compensation, which JACK lacks

Some of these advantages are still being developed. For instance, it is only with the release of Fedora Workstation 35 in October 2021 that improved Bluetooth interaction is one of PipeWire’s features. However, PipeWire is ready for most common purposes.

Resources for Installation and Configuration

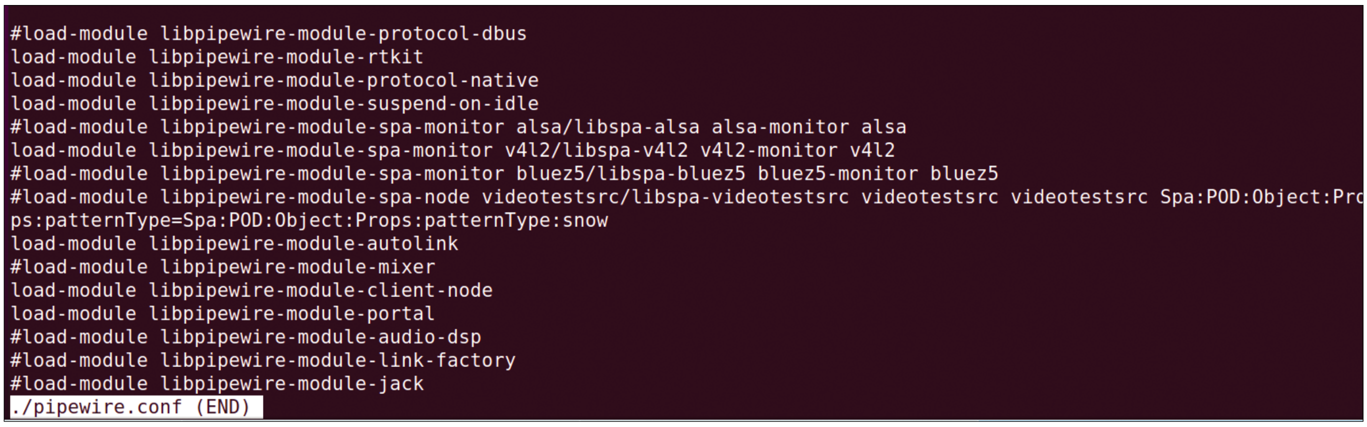
The easiest way to try PipeWire is to install Fedora Workstation 35 in Boxes or VirtualBox. Instructions for working from source code are available online [6] but are only recommended for the hardy. Even other distributions that have PipeWire in their repositories generally have a wiki with distro-specific instructions. Arch Linux [7], Debian [8], and Gentoo [9] are among the distros with PipeWire configuration wikis, although you may find they suffer from a lack of current information. Still, with any luck, you will find that such distro-specific wikis have enough accuracy to install PipeWire both on the distro or on its derivative distros. If you run into trouble, you may find useful information on another distro’s wiki. You can also find information about installing from source, although configuration may be overwhelming unless you are familiar with the common terms used in audio, such as latency, sampling, and buffers, and are willing to take the time to master all the PipeWire components and how they interact with other audio applications and systemd.

The distro-specific PipeWire wikis can be occasionally outdated, but at least you cannot go wrong if you follow the instructions step by step. The Debian wiki is a perfect example. It starts by emphasizing that in Debian 11, the latest release, PipeWire is experimental and may break some applications. Then the wiki goes on to explain the differences in packages between Debian 10 and 11. Clear step-by-step instructions are given for creating the files for replacing PulseAudio with PipeWire in Debian 11, as well as for setting up PipeWire for the root user, if you wish to do that. The same instructions are then given for outputting ALSA and JACK to PipeWire and for using PipeWire with Bluetooth. The Debian wiki then goes on to explain that the PipeWire setup in Debian Testing and Unstable, which will eventually appear in Debian 12, “has vastly improved compatibility and reliability and is also



```
[bb@fedora pipewire]$ ls
client.conf     filter-chain   pipewire.conf
client-rt.conf  jack.conf      pipewire-pulse.conf
```

Figure 1: Fedora stores PipeWire configuration files in /usr/share/pipewire. Not all distributions do the same.



```
#load-module libpipewire-module-protocol-dbus
load-module libpipewire-module-rtkit
load-module libpipewire-module-protocol-native
load-module libpipewire-module-suspend-on-idle
#load-module libpipewire-module-spa-monitor alsa/libspa-alsa alsa-monitor alsa
load-module libpipewire-module-spa-monitor v4l2/libspa-v4l2 v4l2-monitor v4l2
#load-module libpipewire-module-spa-monitor bluez5/libspa-bluez5 bluez5-monitor bluez5
#load-module libpipewire-module-spa-node videotestsrc/libspa-videotestsrc videotestsrc videotestsrc Spa:POD:Object:Pro
ps:patternType=Spa:POD:Object:Props:patternType:snow
load-module libpipewire-module-autolink
#load-module libpipewire-module-mixer
load-module libpipewire-module-client-node
load-module libpipewire-module-portal
#load-module libpipewire-module-audio-dsp
#load-module libpipewire-module-link-factory
#load-module libpipewire-module-jack
./pipewire.conf (END)
```

Figure 2: Ubuntu configures PipeWire from /etc/pipewire/pipewire.conf, but not every distribution does. Ubuntu’s default pipewire.conf lists what is loaded when PipeWire starts.

much easier to configure as a replacement," and also gives more information about special cases. The other wikis are similar, both in instructions and in looking to the future.

Should you want to install PipeWire on different distributions, be aware that PipeWire is so new that it has no standard man page or file placement. The Ubuntu man page has an option to display the version (`--version`) and another option to name the daemon (`--name` or `-n`), neither of which the average user will have much use for. A --help (`-h`) option is available but only lists these three options. The Fedora man page has --verbose (`-v`) and `--config=FILE` (`-c=FILE`) options. However, information is available online for editing the configuration file(s). In Fedora, `/usr/share/pipewire` contains five configuration files, plus another directory of six configuration files (Figure 1). By contrast, Ubuntu uses only `/etc/pipewire/pipewire.conf` (Figure 2). Then, to completely muddle matters, Debian has been using `/etc/pipewire/media-session.d/` but plans to switch the configuration directory to `/usr/share/pipewire/` in Debian 12. For the next few months or so, patience and careful reading are definite pre-requisites for setting up PipeWire.

On the Horizon

Is the effort to set up PipeWire worth it? If you are a casual user of audio on Linux, probably not. After all, it is coming anyway. However, if you are a serious audiophile, then you will appreciate the improvement that PipeWire offers. Sound can be subjective, but even on Boxes, in a virtual installation – hardly the ideal situation – I noticed that the sound on Firefox and when playing FLAC and OGG files was noticeable immediately. In addition, I had no trouble connecting to my Bluetooth speakers. By contrast, every upgrade of my main Debian installation has meant several hours configuring PulseAudio to work consistently with Bluetooth.

Still to come is the use of PipeWire to manage video the same way that it does audio. In the recent buzz, some mention of video capture has been made. However, from what is not said, I can only conclude that the benefits to video are still to come. And what will those be? At this point, we can only wait to see.

Meanwhile, if you have struggled with PulseAudio as much as I have, you should hold on for just a while longer. Help is on the horizon – and about time, too. With any luck, these notes should help you navigate the often inconsistent documentation that has sprung up in PipeWire's wake. ■■■

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (*http://brucebyfield.wordpress.com*). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

Info

[1] Pipewire: *https://pipewire.org/*

[2] "problems with Pulse Audio": *https://www.google.com/search?hl=en&q=problems%20with%20pulseaudio*

[3] "Why do people dislike PulseAudio?": *https://www.reddit.com/r/linux/comments/6h2rvk/why_do_people_dislike_pulseaudio/*

[4] "How Is PipeWire Supposed to Be a Better PulseAudio?": *https://gitlab.freedesktop.org/pipewire/pipewire/-/wikis/FAQ#how-is-pipewire-supposed-to-be-a-better-pulseaudio*

[5] "How Is PipeWire Supposed to Be a Better JACK?": *https://gitlab.freedesktop.org/pipewire/pipewire/-/wikis/FAQ#how-is-pipewire-supposed-to-be-a-better-jack*

[6] Instructions for source code online: *https://docs.pipewire.org/*

[7] Arch Linux: *https://wiki.archlinux.org/title/PipeWire*

[8] Debian: *https://wiki.debian.org/PipeWire*

[9] Gentoo: *https://wiki.gentoo.org/wiki/PipeWire*

Extract and analyze GPS data with Go

Wanderlust

For running statistics on his recorded hiking trails, Mike Schilli turns to Go to extract the GPS data while relying on plotters and APIs for a bit of geoanalysis. *By Mike Schilli*

The GPX data of my hiking trails, which I recorded with the help of geotrackers and apps like Komoot [1], hold some potential for statistical analysis. On which days was I on the move, and when was I lazy? Which regions were my favorites for hiking, and in which regions on the world map did I cover the most miles?

No matter where the GPX files come from – whether recorded by a Garmin tracker or by an app like Komoot that lets you download the data from its website [2] – the recorded data just screams to be put through more or less intelligent analysis programs. For each hike or bike ride, the `tours/` directory (Figure 1) contains one file in XML format (Figure 2). Each of these GPX files consists of a series of geodata recorded with timestamps. In each case, the data shows the longitude and latitude determined using GPS, from which, in turn, you can determine a point on the Earth's surface, visited at a given time.

Nabbed from GitHub

It would be a tedious task to read the XML data manually with Go because its internal structure, with separate tracks, segments, and points, dictates that you have matching structures in Go. Fortunately, someone already solved that problem with the gpxgo project, which is available on GitHub. The program from Listing 1 [3] retrieves it in line 4 and completes the job in one fell swoop using `ParseFile()` in line 26. Any analysis program presented later on just needs to call `gpxPoints()` from Listing 1 with the name of a GPX file to retrieve Go structures with all the geopoints in the file and the matching timestamps.

To calculate the average of all geopoints in a GPX file, say, to determine where the whole trail is located, `gpxAvg()` first calls `gpxPoints()` starting in line 43, uses `pt.Longitude` and `pt.Latitude` from the Point structure to pick up the values for longitude and latitude, and adds them up to create two float64 sums. Also, for each geopoint processed, the `nofPoints` counter is incremented by one, and the averaging function only has to divide the total by the number of points at the end to return the mean value.

Time for an Overview

To get a brief overview of the contents of all collected GPX files, Listing 2 walks through all the files in the `tours/` directory using `gpxFiles()` from Listing 1. It reads the files' XML data and uses `gpxPoints()` to return a list of all the geopoints it contains along with matching timestamps.

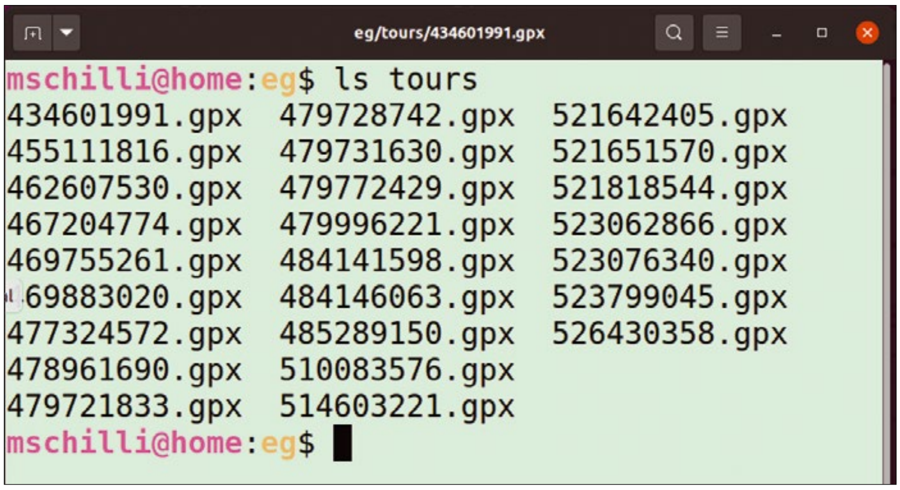
The output in Figure 3 shows that the trails were recorded all over the place. For example, the intersection of the latitude of 37 degrees north and the longitude of -122 degrees west is my adopted home of San Francisco. On the other hand, the latitude of 48 degrees north and the longitude of 10 degrees east represents my former home of Augsburg, Germany, which I visited as an American tourist last summer.

Out and About or Lazy?

A recording's GPX points also come with timestamps; in other words, a collection of GPX files reveals the calendar

Author

Mike Schilli works as a software engineer in the San Francisco Bay Area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at *mschilli@perlmeister.com* he will gladly answer any questions.



```
mschilli@home:eg$ ls tours
434601991.gpx  479728742.gpx  521642405.gpx
455111816.gpx  479731630.gpx  521651570.gpx
462607530.gpx  479772429.gpx  521818544.gpx
467204774.gpx  479996221.gpx  523062866.gpx
469755261.gpx  484141598.gpx  523076340.gpx
469883020.gpx  484146063.gpx  523799045.gpx
477324572.gpx  485289150.gpx  526430358.gpx
478961690.gpx  510083576.gpx
479721833.gpx  514603221.gpx
mschilli@home:eg$ 
```

Figure 1: Homegrown analysis tools extract motion data from a collection of GPX files.

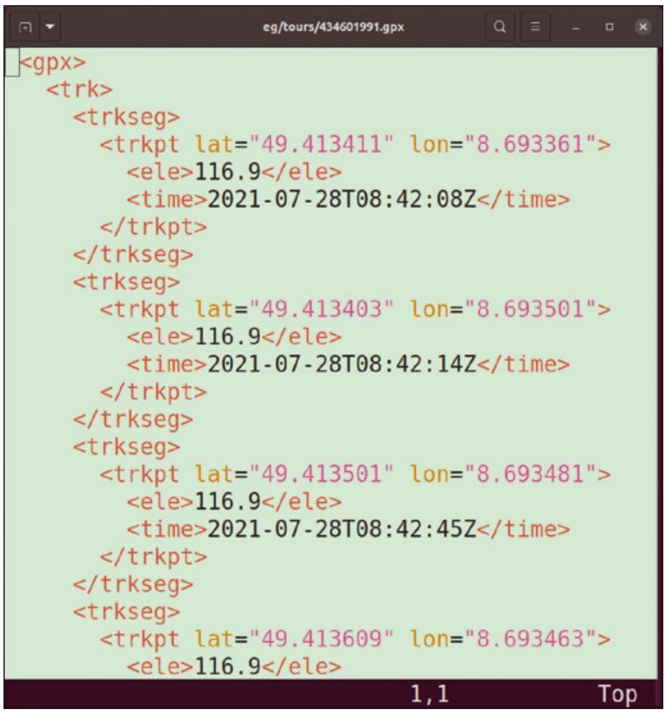
Photo by Martin Kallur (IG: @mkallur) on Unsplash

Listing 1: gpxread.go

```
package main

import (
  "github.com/tkrajina/gpxgo/gpx"
  "os"
  "path/filepath"
)

func gpxFiles() []string {
  tourDir := "tours"
  files := []string{}

  entries, err := os.ReadDir(tourDir)
  if err != nil {
    panic(err)
  }

  for _, entry := range entries {
    gpxPath := filepath.Join(tourDir, entry.Name())
    files = append(files, gpxPath)
  }
  return files
}

func gpxPoints(path string) []gpx.GPXPoint {
  gpxData, err := gpx.ParseFile(path)
  points := []gpx.GPXPoint{}

  if err != nil {
    panic(err)
  }

  for _, trk := range gpxData.Tracks {
    for _, seg := range trk.Segments {
      for _, pt := range seg.Points {
        points = append(points, pt)
      }
    }
  }
  return points
}

func gpxAvg(path string)(float64, float64, int) {
    nofPoints := 0
    latSum,longSum := 0.0, 0.0
    for _, pt := range gpxPoints(path) {
      latSum += pt.Latitude
      longSum += pt.Longitude
      nofPoints++
    }
    return latSum/float64(nofPoints),
      longSum/float64(nofPoints), nofPoints
 }
```

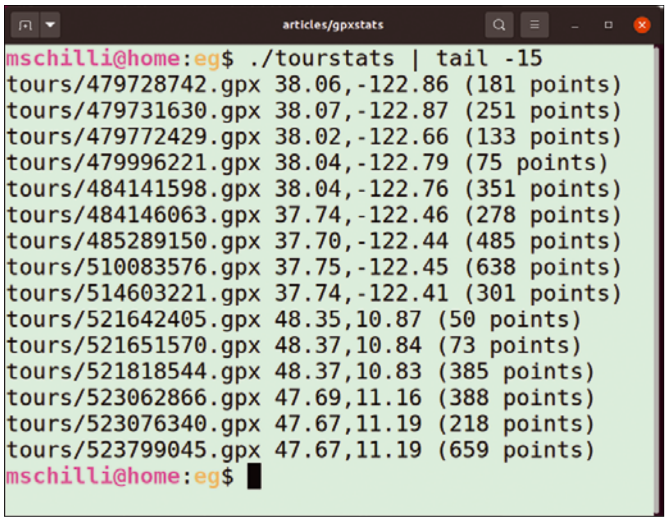
Listing 2: tourstats.go

```
package main

import ("fmt")

func main() {
  for _, path := range gpxFiles() {
    lat, lon, pts := gpxAvg(path)
    fmt.Printf("%s %.2f,%.2f (%d points)\n",
      path, lat, lon, pts)
  }
}
```



```
<gpx>
  <trk>
    <trkseg>
      <trkpt lat="49.413411" lon="8.693361">
        <ele>116.9</ele>
        <time>2021-07-28T08:42:08Z</time>
      </trkpt>
    </trkseg>
    <trkseg>
      <trkpt lat="49.413403" lon="8.693501">
        <ele>116.9</ele>
        <time>2021-07-28T08:42:14Z</time>
      </trkpt>
    </trkseg>
    <trkseg>
      <trkpt lat="49.413501" lon="8.693481">
        <ele>116.9</ele>
        <time>2021-07-28T08:42:45Z</time>
      </trkpt>
    </trkseg>
    <trkseg>
      <trkpt lat="49.413609" lon="8.693463">
        <ele>116.9</ele>
                                      1,1           Top
```

Figure 2: The XML data in the GPX file represents track points on a hiking trail.



```
mschilli@home:eg$ ./tourstats | tail -15
tours/479728742.gpx 38.06,-122.86 (181 points)
tours/479731630.gpx 38.07,-122.87 (251 points)
tours/479772429.gpx 38.02,-122.66 (133 points)
tours/479996221.gpx 38.04,-122.79 (75 points)
tours/484141598.gpx 38.04,-122.76 (351 points)
tours/484146063.gpx 37.74,-122.46 (278 points)
tours/485289150.gpx 37.70,-122.44 (485 points)
tours/510083576.gpx 37.75,-122.45 (638 points)
tours/514603221.gpx 37.74,-122.41 (301 points)
tours/521642405.gpx 48.35,10.87 (50 points)
tours/521651570.gpx 48.37,10.84 (73 points)
tours/521818544.gpx 48.37,10.83 (385 points)
tours/523062866.gpx 47.69,11.16 (388 points)
tours/523076340.gpx 47.67,11.19 (218 points)
tours/523799045.gpx 47.67,11.19 (659 points)
mschilli@home:eg$ █
```

Figure 3: A collection of GPX files and their analysis by `tourstats.go`.

Listing 3: activity.go

```
package main

import (
  "fmt"
  "github.com/wcharczuk/go-chart/v2"
  "os"
  "sort"
  "time"
)

func main() {
  perday := map[time.Time]int{}

  for _, path := range gpxFiles() {
    for _, pt := range gpxPoints(path) {
      t := time.Date(pt.Timestamp.Year(), pt.Timestamp.
                     Month(), pt.Timestamp.Day(), 0, 0,
                     0, 0, time.Local)
      perday[t]++
    }
  }

  keys := []time.Time{}
  for day, _ := range perday {
    keys = append(keys, day)
  }
  sort.Slice(keys, func(i, j int) bool {
    return keys[i].Before(keys[j])
  })

  xVals := []time.Time{}
  yVals := []float64{}
  for _, key := range keys {
    xVals = append(xVals, key)
    yVals = append(yVals, float64(perday[key]))
  }

  mainSeries := chart.TimeSeries{
    Name: "GPS Activity",
    Style: chart.Style{
      StrokeColor: chart.ColorBlue,
      FillColor: chart.ColorBlue.WithAlpha(100),
    },
    XValues: xVals,
    YValues: yVals,
  }

  graph := chart.Chart{
    Width:  1280,
    Height: 720,
    Series: []chart.Series{mainSeries},
  }

  f, _ := os.Create("activity.png")
  defer f.Close()

  graph.Render(chart.PNG, f)
}
```

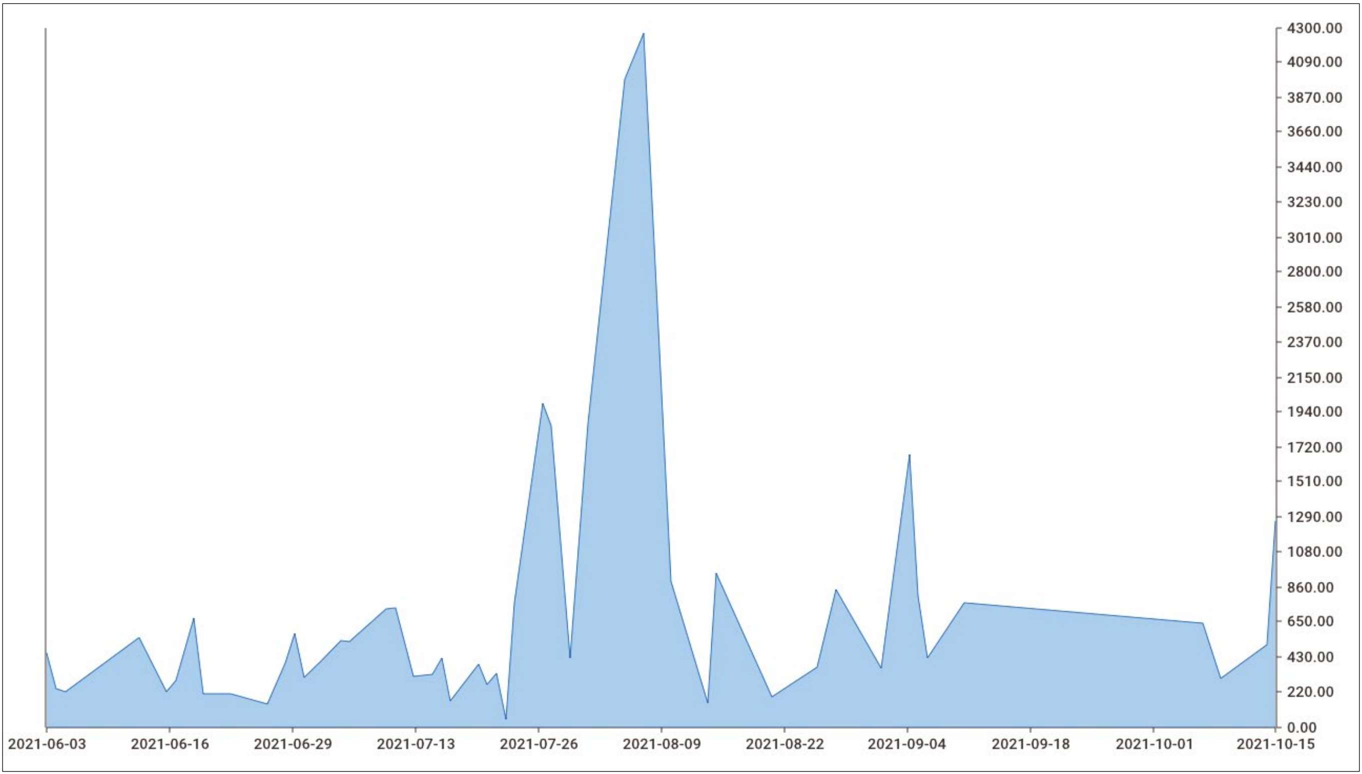


Figure 4: **Walking activities as a number of GPS track points over time.**

days on which I recorded walks. From this data, Listing 3 generates a time-based activity curve. To accumulate the number of all track points recorded during a given calendar day, it sets the hour, minute, and second values of all timestamps it finds to zero and uses `time.Date()` to set the recording date, valid for all points sampled during a specific calendar day. In the `perday` hash map, line 17 then increments the respective day entry by one with each matching timestamp it finds. All that remains to do then is to sort the keys of the hash map (i.e., the date values) in ascending order and to draw them on a chart with values corresponding to the assigned counters.

Simply Complicated

This is by no means as simple as in scripting languages such as Python, where hashes are also unsorted, but a `sort` command on the hash keys quickly returns them in order, for a loop to process the entries. In Go, the `for` loop starting in line 22 of Listing 3 first needs to collect all the keys in the hash map and push them into a newly created array slice. The `sort.Slice()` function then sorts the slice by time in ascending order, starting in line 25.

With an array slice of strings, you could do the whole thing quickly with `sort.Strings()`, but because the hash keys are `time.Time` type data, line 26 still has to define a callback function that tells the sort algorithm which of two array values at indexes `i` versus `j` is now larger. Fortunately, the `time.Time` type has a `Before()` function that returns exactly this result, so the callback function simply returns the result of that, because that's exactly what the comparison function is supposed to determine.

With the keys sorted by time in ascending order, the `for` loop starting in line 31 can now pass the hash entries, also sorted in a separate array slice, to the `go-chart` plotting library that line 5 in Listing 3 pulls from GitHub. It draws visually appealing charts as bar diagrams, pie charts, or function graphs. In the case at hand, the data exists as a series of values

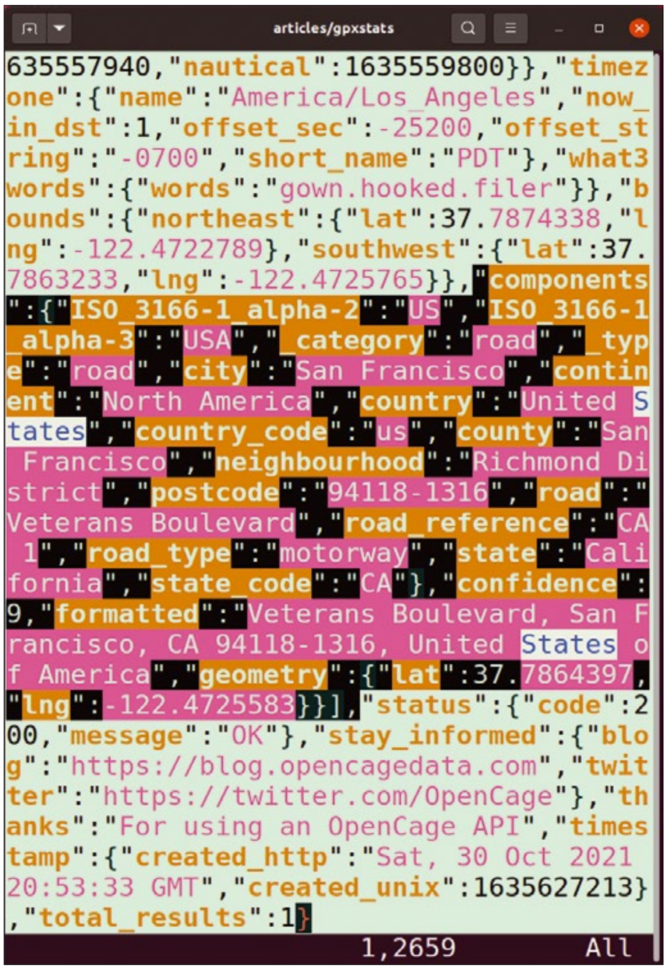
Listing 4: georev.go

```
package main

import (
  "encoding/json"
  "fmt"
  "github.com/peterbourgon/diskv"
  "io/ioutil"
  "net/http"
  "net/url"
)

type GeoState struct {
  ApiKey string
  Cache  *diskv.Diskv
}

func NewGeoState() *GeoState {
  state := GeoState{
    ApiKey: "<API-Key>",
    Cache:  diskv.New(diskv.Options{BasePath: "cache"}),
  }
  return &state
}

func (state *GeoState) GeoRev(lat, lng float64) string {
  key := roundedLatLng(lat, lng)

  res, err := state.Cache.Read(key)
  if err != nil {
    res = []byte(state.GeoLookup(lat, lng))
    state.Cache.Write(key, res)
  }

  return string(res)
}

func roundedLatLng(lat, lng float64) string {
  return fmt.Sprintf("%.1f,%.1f", lat, lng)
}

func (state *GeoState) GeoLookup(lat, lng float64) string {
  u := url.URL{
    Scheme: "https",
    Host:   "api.opencagedata.com",
    Path:   "geocode/v1/json",
  }
  q := u.Query()
  q.Set("key", state.ApiKey)
  q.Set("q", roundedLatLng(lat, lng))
  u.RawQuery = q.Encode()

  resp, err := http.Get(u.String())
  if err != nil {
    panic(err)
  }

  body, err := ioutil.ReadAll(resp.Body)
  if err != nil {
    panic(err)
  }
  return stateFromJson(body)
}

func stateFromJson(txt []byte) string {
  var data map[string]interface{}
  json.Unmarshal(txt, &data)

  results := data["results"].([]interface{})[0].
    (map[string]interface{})
  return results["components"].(map[string]interface{})
    ["state"].(string)
}
```

at specific time points, so line 36 defines the chart as `chart.TimeSeries` and sets the fill and stroke color to blue. You can compile the program with

```
go build activity.go gpxread.go
```

because it loads in the `gpxFiles()` and `gpxPoints()` functions from Listing 1 to support its plot instructions. The output it generates is an image file named `activity.png` (Figure 4).



```
635557940,"nautical":1635559800}},"timez
one":{"name":"America/Los_Angeles","now_
in_dst":1,"offset_sec":-25200,"offset_st
ring":"-0700","short_name":"PDT"},"what3
words":{"words":"gown.hooked.filer"}},"b
ounds":{"northeast":{"lat":37.7874338,"l
ng":-122.4722789},"southwest":{"lat":37.
7863233,"lng":-122.4725765}},"components
":{"ISO_3166-1_alpha-2":"US","ISO_3166-1
_alpha-3":"USA","_category":"road","_typ
e":"road","city":"San Francisco","contin
ent":"North America","country":"United S
tates","country_code":"us","county":"San
 Francisco","neighbourhood":"Richmond Di
strict","postcode":"94118-1316","road":"
Veterans Boulevard","road_reference":"CA
 1","road_type":"motorway","state":"Cali
fornia","state_code":"CA"},"confidence":
9,"formatted":"Veterans Boulevard, San F
rancisco, CA 94118-1316, United States o
f America","geometry":{"lat":37.7864397,
"lng":-122.4725583}}],"status":{"code":2
00,"message":"OK"},"stay_informed":{"blo
g":"https://blog.opencagedata.com","twit
ter":"https://twitter.com/OpenCage"},"th
anks":"For using an OpenCage API","times
tamp":{"created_http":"Sat, 30 Oct 2021
20:53:33 GMT","created_unix":1635627213}
,"total_results":1}
                      1,2659         All
```

Figure 5: The JSON response to geomapping from OpenCage contains detailed location information.

Real World

But given the GPX coordinates in geographical longitude and latitude, how do you discover the country in which they are located, or even the state, city, or street? This function is the domain of geomapping tools. This is actually a reverse case – it's not a matter of inferring the geolocation from a given address but discovering the city from the given coordinates. To do this, you ultimately need a huge database that assigns places on the Earth's map to their GPS coordinates in as much detail as possible.

To perform this calculation, there are various services online vying for the favor of their paying customers, who are then allowed to access their treasure trove of data via API. Google Maps originally offered this kind of mapping free of charge, but the boffins at Google withdrew this offer some time ago and now require a credit card to register. The company then bills you if you exceed a certain number of requests.

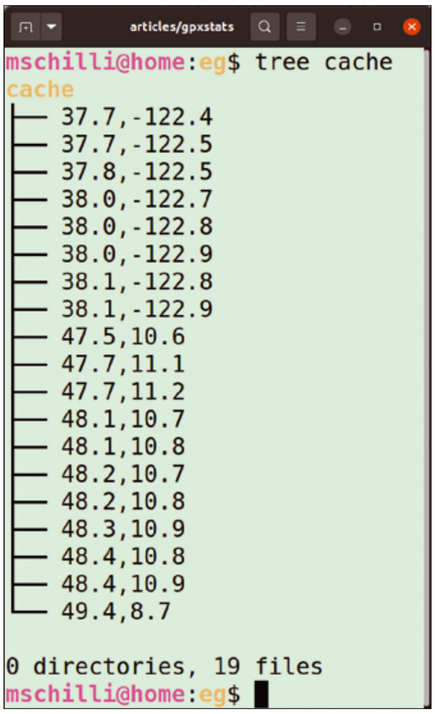
But if you look around online, you will find a number of freemium offers from some providers such as OpenCage [4], where you can register by email and receive an API token for a limited number of requests for a trial. As an example of using the API, the `GeoRev()` function, starting in line 25 of Listing 4, converts a combination of the latitude and longitude in float64 format to the state at that location. This means that `37.7, -122.4` becomes `California` and `49.4, 8.7` becomes `Bavaria`.

Figure 5 shows the detailed JSON response provided by the OpenCage server for a coordinate in the San Francisco metropolitan area. Everything from the zip code to the street name, country, state, and so on is shown there. Of course, this only works if Listing 4 (in line 19) contains a valid API key, which is available on the OpenCage site if you register your email (no credit card required).

Listing 5: states.go

```
package main

import (
  "github.com/wcharczuk/go-chart/v2"
  "os"
)

func main() {
  geo := NewGeoState()
  perState := map[string]int{}

  for _, path := range gpxFiles() {
    for _, pt := range gpxPoints(path) {
      state := geo.GeoRev(pt.Latitude, pt.Longitude)
      perState[state]++
    }
  }

  vals := []chart.Value{}
  for state, count := range perState {
    vals = append(vals, chart.Value{Value: float64(count),
              Label: state})
  }

  pie := chart.PieChart{
    Width:  512, Height: 512,
    Values: vals,
  }

  f, _ := os.Create("states.png")
  defer f.Close()
  pie.Render(chart.PNG, f)
}
```

Using Go to extract data from a server response with JSON data is always difficult if there is no corresponding Go type on the client side that can reproduce the data structure down to the last detail. The alternative is a hack that relies on type assertion to force the data into hash maps with `interface{}` types. The `stateFromJson()` function works its way through the `results`, `components`, and `state` entries in the JSON data before it finds the nugget it's looking for, while Go uses type assertions to constantly assure the function that the next part really is of the expected type.



```
mschilli@home:eg$ tree cache
cache
├── 37.7,-122.4
├── 37.7,-122.5
├── 37.8,-122.5
├── 38.0,-122.7
├── 38.0,-122.8
├── 38.0,-122.9
├── 38.1,-122.8
├── 38.1,-122.9
├── 47.5,10.6
├── 47.7,11.1
├── 47.7,11.2
├── 48.1,10.7
├── 48.1,10.8
├── 48.2,10.7
├── 48.2,10.8
├── 48.3,10.9
├── 48.4,10.8
├── 48.4,10.9
└── 49.4,8.7

0 directories, 19 files
mschilli@home:eg$
```

Figure 6: The `cache` directory contains the results of reverse geomappings that have already been retrieved.

Smart Cache

Of course, it would be pretty stupid to bomb the server with thousands of requests for almost identical track points – on the one hand, because this requires a round trip across the Internet each time, and on the other, because the limited quota of the free trial account would be quickly consumed.

This is why Listing 4 defines a `cache/` directory, where the simple `discv` cache implementation from GitHub permanently caches previously retrieved API results for later use. The `rounded-LatLng()` function, starting in line 37, rounds the longitude and latitude to one decimal place for incoming requests and first looks to see if there is already a result in the cache. If not, it fetches the corresponding geodata with a web request to the server's API and stores what it needs from it in the cache. While the program is running, the cache fills up visibly with the incoming requests, as a quick look at the `cache` directory in Figure 6 reveals.

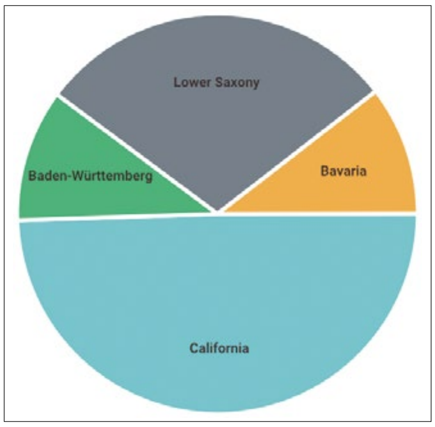


Figure 7: U.S. and German states visited on recorded hikes.

Armed with this package, Listing 5 can now draw a pie chart indicating which states the collection of GPX files cover in total and where the most hiking steps were recorded. To do this, in the `perState` hash map, it increments the entry under the state string key by one. The more track points there are in the respective region, the higher the count value, and the bigger the pie slice in the graph later on.

Listing 5 generates an image file named `states.png`, and Figure 7 shows the results: My GPX files are mostly from California but also feature the German states of Bavaria, Baden-Württemberg, and Lower Saxony.

I Want More

Of course, these are only crude examples that show what is possible. With a little more effort, more hidden nuggets can be extracted from the data. For example, if you want to identify the regions with the most popular trails, you can use an AI library to track down clusters using the k-means method. In addition, artificial intelligence could suggest previously popular hiking routes that you have not visited for some time. And there's so much more; once the data is recorded, the world is your oyster. ■■■

Info

[1] Komoot: *https://www.komoot.com/*

[2] "Programming Snapshot: Go GPS Data Retrieval" by Mike Schilli, *Linux Magazine*, issue 252, November 2021, p. 50

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/256/*

[4] OpenCage: *https://opencagedata.com*

Packaging applications in Docker containers

Neatly Bundled

Kaboxer lets users deploy applications that may be difficult to package using Docker containers.

By Ferdinand Thommes

The way distributions deliver software is changing. New package formats such as Flatpak and Snap are becoming more widespread, and containers are becoming increasingly important even for casual desktop users. There are many reasons for this. Developers want to see their software quickly reach users without having to create packages in different formats. Some approaches also allow multiple software versions to be installed simultaneously. Sandboxing as a security feature also plays a role.

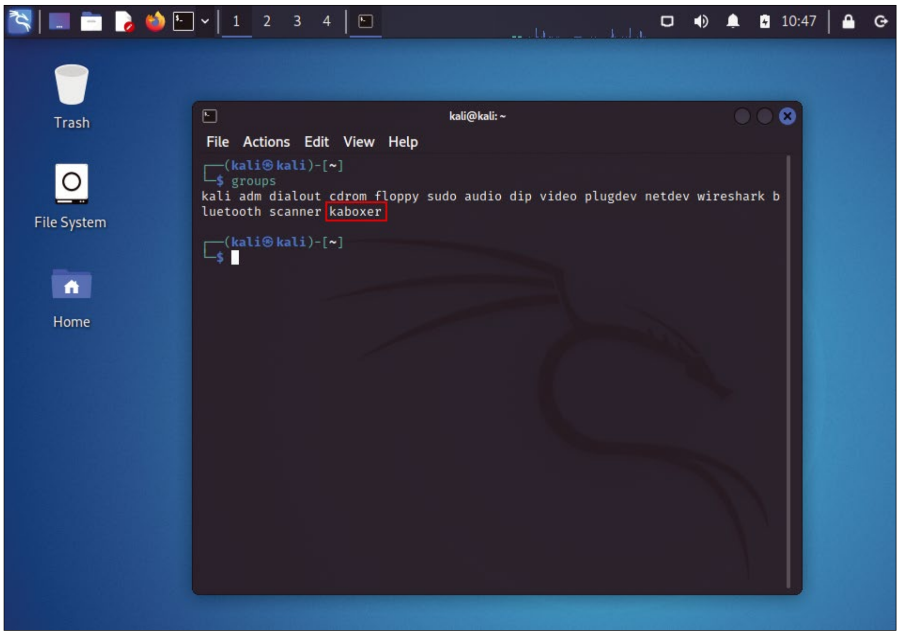
In addition, not all software can be easily packaged and kept up-to-date using traditional package formats. This is especially true for distributions such as Kali Linux which ship hundreds of highly specialized applications. Many of these specialized applications are unavailable in the Debian repository. Others are difficult to package because they rely on outdated libraries that hardly any distributions come with anymore. Another reason would be to isolate apps so that they do not interfere with other programs.

Kaboxer [1], a Docker and DEB package-based application developed for Kali Linux, transparently deploys difficult-to-package applications in Docker containers within the Debian packaging system.

Kaboxer

Kali Linux specializes in penetration tests and digital forensics. Based on Debian, Kali Linux uses the Debian package manager. Kaboxer (an abbreviation of Kali Applications Boxer) extends the Debian package system via containers but integrates them into the existing system and controls them transparently via Kaboxer.

The Kaboxer developers emphasize the compatibility of this approach with other Debian variants in the documenta-



```
┌──(kali㉿kali)-[~]
└─$ groups
kali adm dialout cdrom floppy sudo audio dip video plugdev netdev wireshark b
luetooth scanner kaboxer

┌──(kali㉿kali)-[~]
└─$ 
```

Figure 1: **If you want to use the container framework, make sure that your user account belongs to the *kaboxer* group.**

Listing 1: Kaboxer Hands-On

```
$ sudo apt install kaboxer
$ apt search --names-only '\-kbx$'
$ sudo apt install zenmap-kbx
$ sudo apt install covenant-kbx
```

tion. The developers create Docker images of the applications, which they link in classic Debian packages. During installation, these packages then download the images. To create the DEBs, the Kaboxer team has extended Debian's packaging tool debhelper to include a `debhelper_kaboxer` option and adapted the build system to match. As a user, you install the packages in the normal way with:

```
sudo apt install
```

Afterward, you will find the applications in the main menu.

Docker Makes It Possible

The Kaboxer developers' decision to use Docker does not exclude other container formats in the future. Docker was initially chosen because its containers come with a large number of parameters for configuration, which means that the images can be easily integrated, both with the host system and across multiple containers.

To ensure this integration, Kaboxer uses existing Docker features such as mount points and port redirects. Menu items are based on `.desktop` files created by Kaboxer. All the integration details, as well as the instructions for creating or retrieving the Docker image, are bundled in a single YAML file. The file, in turn, is packaged in one of the DEB files provided by the Kali project. The `post-inst` script for these packages downloads the image so that the application it contains can be used immediately afterwards.

Transparently Integrated

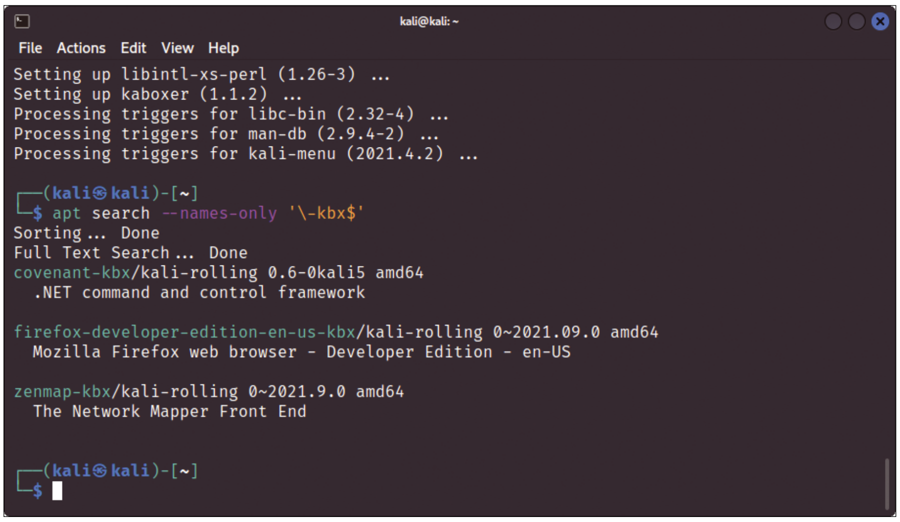
After containerizing an app, Kaboxer's next task is to deploy the app so that users can open it with the familiar Debian package management commands. Kaboxer's other tasks include ensuring the persistence of the data created by the user with the respective application, even if the user deletes the corresponding container.

This explains why Kaboxer comes with functions for configuring volumes shared between the host and the container. Additional steps need to be taken for GUI or web applications: GUI applications, for example, need access to the host's X11 socket. For web applications, the HTTP port must be allowed, and the web browser must be launched with the respective URL.

Easy as Pie

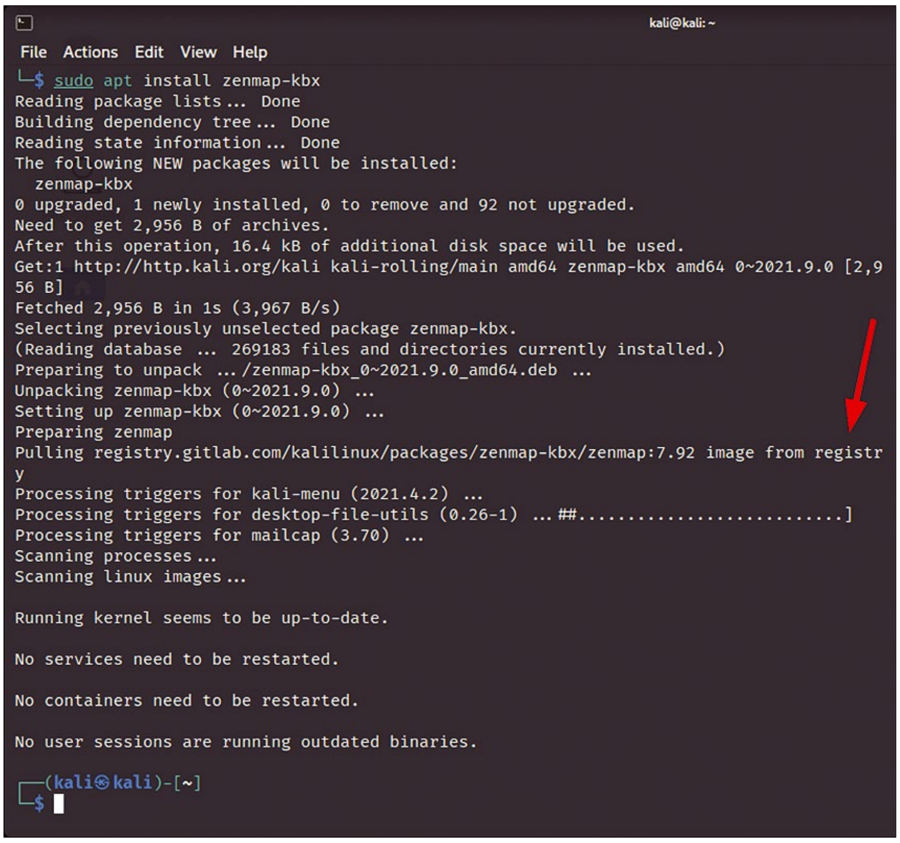
Currently, the Kaboxer developers have packaged three applications. I'll take a look at these applications from a user's perspective by installing a GUI application and a web application.

To recreate the examples, first install the main `kaboxer` tool (Listing 1, line 1). Then make sure that your user account belongs to the *kaboxer* group (Figure 1). If you



```
Setting up libintl-xs-perl (1.26-3) ...
Setting up kaboxer (1.1.2) ...
Processing triggers for libc-bin (2.32-4) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.4.2) ...

┌──(kali㉿kali)-[~]
└─$ apt search --names-only '\-kbx$'
Sorting... Done
Full Text Search... Done
covenant-kbx/kali-rolling 0.6-0kali5 amd64
  .NET command and control framework

firefox-developer-edition-en-us-kbx/kali-rolling 0~2021.09.0 amd64
  Mozilla Firefox web browser - Developer Edition - en-US

zenmap-kbx/kali-rolling 0~2021.9.0 amd64
  The Network Mapper Front End


┌──(kali㉿kali)-[~]
└─$ 
```

Figure 2: With a search via `apt`, you can quickly determine which applications are in Kaboxer format.



```
└─$ sudo apt install zenmap-kbx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  zenmap-kbx
0 upgraded, 1 newly installed, 0 to remove and 92 not upgraded.
Need to get 2,956 B of archives.
After this operation, 16.4 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 zenmap-kbx amd64 0~2021.9.0 [2,9
56 B]
Fetched 2,956 B in 1s (3,967 B/s)
Selecting previously unselected package zenmap-kbx.
(Reading database ... 269183 files and directories currently installed.)
Preparing to unpack .../zenmap-kbx_0~2021.9.0_amd64.deb ...
Unpacking zenmap-kbx (0~2021.9.0) ...
Setting up zenmap-kbx (0~2021.9.0) ...
Preparing zenmap
Pulling registry.gitlab.com/kalilinux/packages/zenmap-kbx/zenmap:7.92 image from registr
y
Processing triggers for kali-menu (2021.4.2) ...
Processing triggers for desktop-file-utils (0.26-1) ...##..........................]
Processing triggers for mailcap (3.70) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

┌──(kali㉿kali)-[~]
└─$ 
```

Figure 3: When installing Kaboxer packages via Debian's package system, the actual contents enter the system as a Docker image from the Kali developers' GitLab registry.

want to know which applications are available in Kaboxer format as well as which ones you have installed, use the command from Listing 1, line 2 (Figure 2).

First, I will test Zenmap, a graphical front end for the Nmap port scanner. Zenmap depends on deprecated Python 2 libraries, which makes any kind of normal software installation on Debian and its derivatives painful because Python 2 has been removed from the Debian repositories.

To set up Zenmap with less overhead, the Kali developers offer the *zenmap-kbx* package (Listing 1, line 3) as one of their first Kaboxer applications. Figure 3 shows the installation process, which also creates an entry in the desktop menu structure (Figure 4). You can set up Covenant, a .NET-based command-and-control framework including a web front end (Figure 5) in the same way using the *covenant-kbx* package (line 4).

Both apps work immediately. You will not even notice that they are running in containers. Covenant automatically launches its web interface. The Kaboxer version of Firefox as a full-fledged GUI application also performed without any issues in testing.

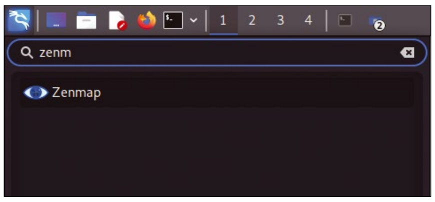


Figure 4: Kaboxer automatically creates an entry in the main desktop menu from which you start the container.

Disadvantages

Kaboxer apps take a little longer to install than traditional DEB packages. A Kaboxer application's tandem packages essentially function as a metapackage that downloads the content as a Docker image and then sets it up. However, you will not notice any further delays once you launch these apps.

The Kaboxer model has a disadvantage also common to Flatpak and Snap: file size. Even programs that are only a few kilobytes tend to swell to 50MB or more as Kaboxer containers due to the unavailable dependencies on the host system (or dependencies that might even be duplicated in the worst case scenario), as well as the container's overhead. Because of this, Kali Linux does not include these applications in its operating system images. Instead, you must install them manually if needed.

DIY

Setting up applications for Kaboxer is definitely not witchcraft. The tool's documentation describes the required steps in detail [2]. The steps include creating and building a Docker image of the corresponding application. For larger-scale deployments, it would make sense to automate the process via GitLab CI. For more information beyond what is described in the documentation, see the Kaboxer man pages and `kaboxer.yaml`.

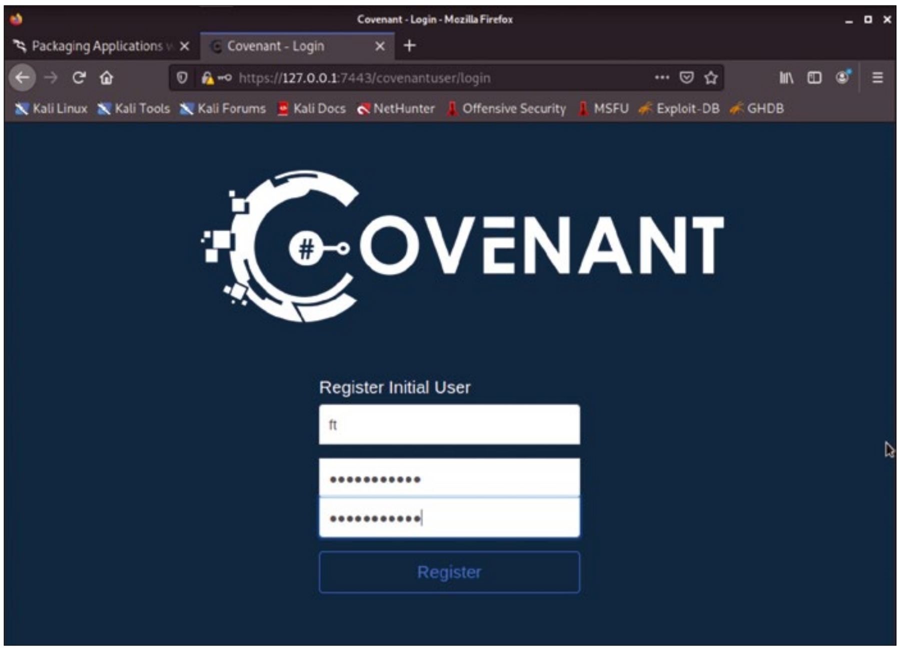


Figure 5: Covenant requires a bit more effort because you operate it via a web interface that the container launches in the browser.

Conclusions and Outlook

For Kali Linux users, Kaboxer offers the advantage of installing tools and applications that would otherwise be difficult or impossible to set up. You may notice (if at all) the use of containers instead of regular DEB packages due to the somewhat longer installation time. With a little practice, you can package simple applications quite quickly. Whether the developers of other Debian derivatives will eventually take advantage of Kaboxer's technology remains to be seen. ∎∎∎

Info

[1] Kaboxer: *https://gitlab.com/kalilinux/tools/kaboxer*

[2] Documentation: *https://www.kali.org/docs/development/packaging-apps-with-kaboxer/*

FOSSLIFE

Learn what it's all about

FOSSlife.org

MakerSpace

Industrial control programming and protocols on a Raspberry Pi

Small Industry

Create automation projects with ladder logic, function blocks, structured text, and Modbus TCP. *By Pete Metcalfe*

Home automation projects can take advantage of a number of good software packages (e.g., Home Assistant and Node-RED) to manage and monitor sensors and controllable devices. If you are interested in looking at an industrial controls approach to your automation projects, then OpenPLC [1] is a good package to consider. A programmable logic controller (PLC) is a hardened industrial hardware device that manages I/O and logic according to the IEC 61131-3 standard [2].

The OpenPLC open source software runs on a Raspberry Pi, Linux, or Windows PC, and it offers users a great way to learn industrial control concepts, programming languages, and communications protocols.

In this article, I introduce IEC 61131-3 programming by creating three small OpenPLC programs that use ladder logic, function blocks, and structured text programming languages. The programs in the projects presented here connect to the Raspberry Pi general purpose input/output (GPIO) pins for basic read and

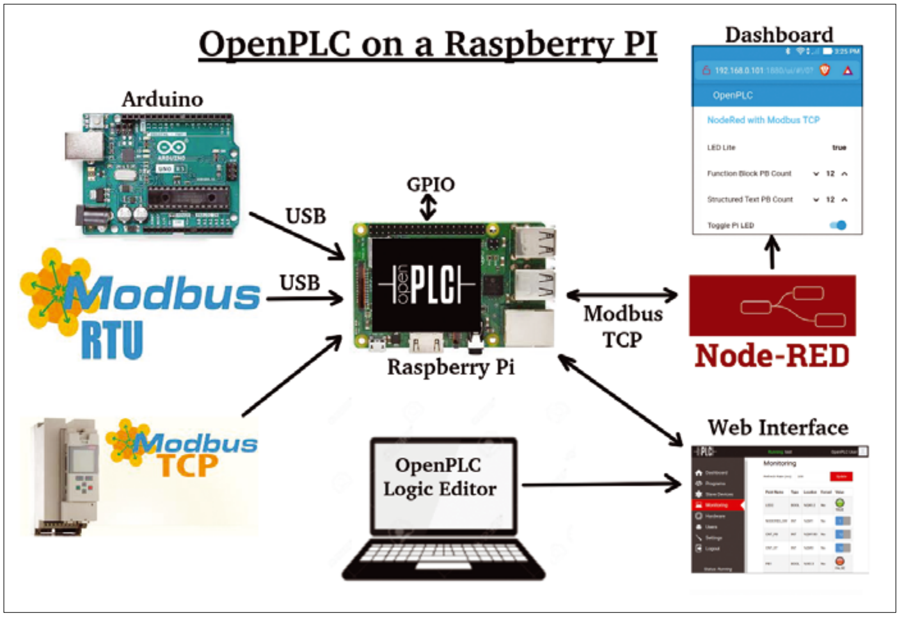
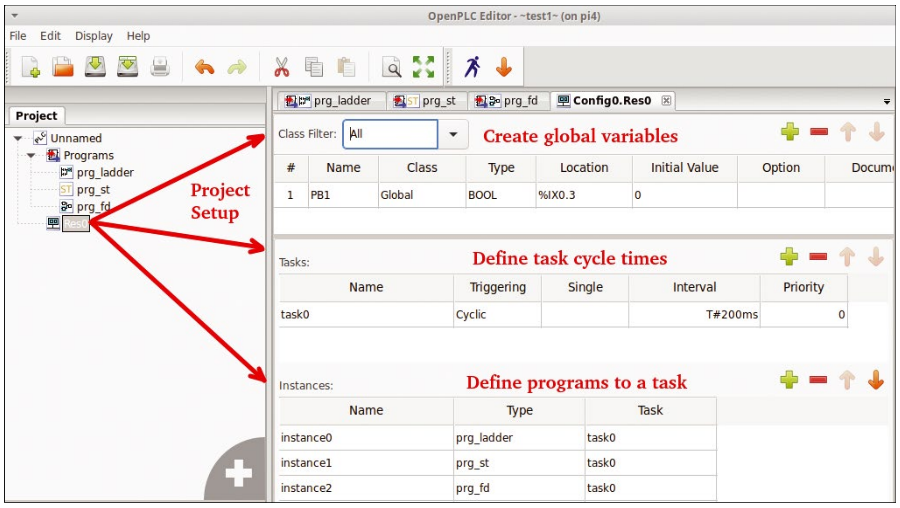


Figure 1: OpenPLC on a Raspberry Pi.



OpenPLC Editor - ~test1~ (on pi4)

File Edit Display Help

Project

- Unnamed
  - Programs
    - prg_ladder
    - prg_st
    - prg_fd
  - Res0

Project Setup

prg_ladder | prg_st | prg_fd | Config0.Res0

Class Filter: All

Create global variables

#	Name	Class	Type	Location	Initial Value	Option	Docum
1	PB1	Global	BOOL	%IX0.3	0		

Tasks:

Define task cycle times

Name	Triggering	Single	Interval	Priority
task0	Cyclic		T#200ms	0

Instances:

Define programs to a task

Name	Type	Task
instance0	prg_ladder	task0
instance1	prg_st	task0
instance2	prg_fd	task0

Figure 2: OpenPLC project setup.

write functions; then, the PLC project passes data by Modbus TCP to a Node-RED [3] dashboard.

Getting Started

The OpenPLC software comes in three packages: a logic editor, the runtime component, and a graphic builder. You can find specific instructions for your installation online [4]. For my installation, I put the OpenPLC editor on my Ubuntu PC for some standalone configuration and testing, but you could also load both the editor and OpenPLC runtime on a Raspberry PI. The OpenPLC runtime has a web interface, so logic can be uploaded through a web browser. I didn't install the OpenPLC graphic builder; instead, I used Node-RED dashboards as my final user interface.

OpenPLC has a good number of optional communications packages and subsidiary I/O components (Figure 1). For this application, I created an OpenPLC project with three programs: a ladder program, a function block program, and a structured text program.

The resource object (Res0) defines global variables that can be used by all programs (e.g., pushbutton PB1) and the task cycle times (Figure 2). One of the benefits of a PLC is that it allows for easy management of task scheduling and program cycle times. Because this project is small, I put all the programs into the same task execution (task0). For a larger project, I might put all my digital logic into a fast task execution (20ms) and my analog logic into a slower task execution (250ms).

I wanted OpenPLC to do some basic Raspberry Pi pin reads and writes, so I used a setup with a pushbutton on pin 17 and an LED on pin 23 (Figure 3). OpenPLC defines the Raspberry Pi GPIO pins by IEC 61131-3 addressing (Figure 4). For this project, the pushbutton at Broadcom SOC channel (BCM) pin 17 (physical pin 11) is addressed by %IX0.3, which means an input bit on bus 0 at bit 3. The LED at BCM pin 23 (physical pin 16) is addressed by %QX0.2, an output bit on bus 0 bit 2.

Note that OpenPLC has allocated all the left side (odd) pins of the Raspberry Pi as inputs and all the right side (even) pins as outputs.

The first step in this project is to create some IEC 61131-3 programs to connect to the Pi pushbutton and LED. Once the programs are created, they are loaded and compiled on the OpenPLC runtime; the web interface

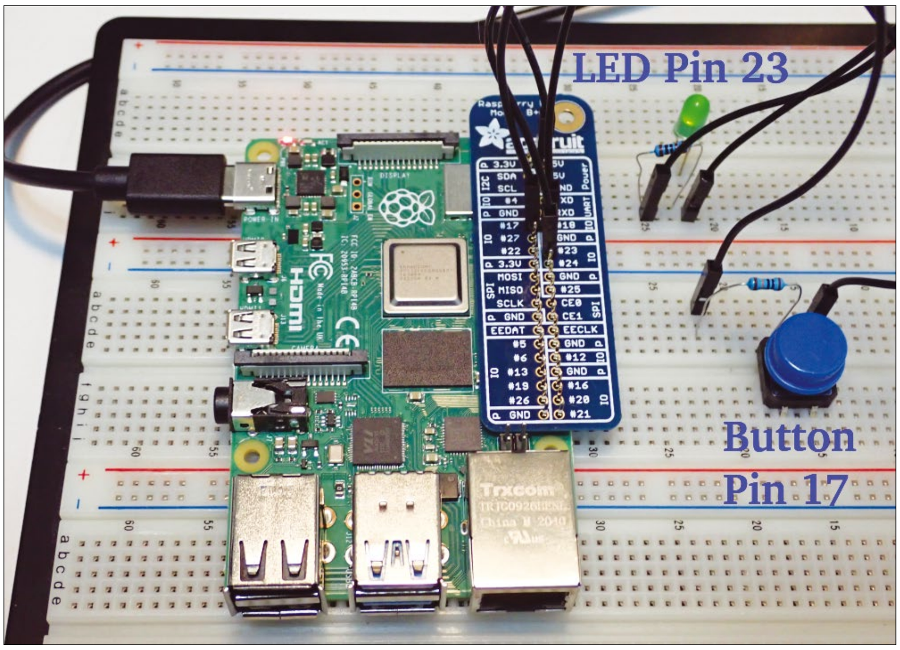


Figure 3: Raspberry Pi hardware setup.

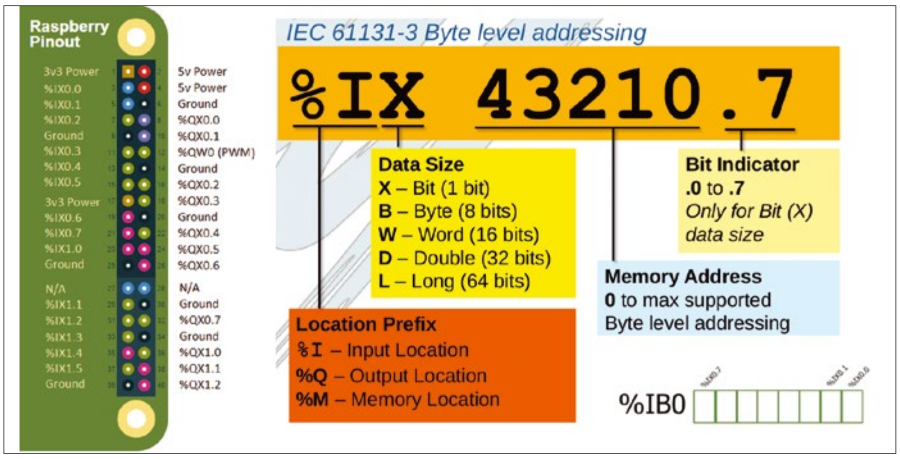


Figure 4: IEC 61131-3 addressing for Raspberry Pi pins.

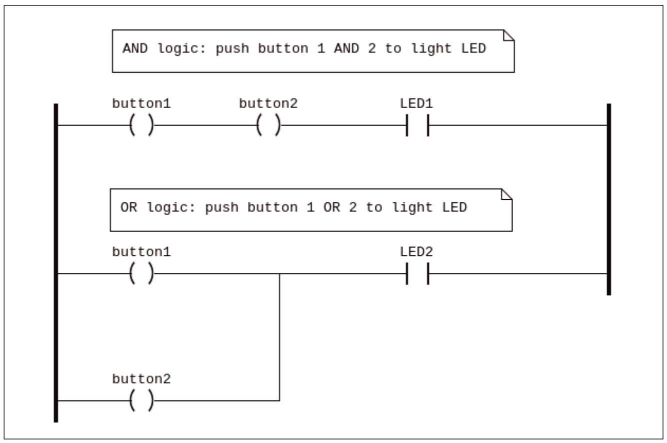
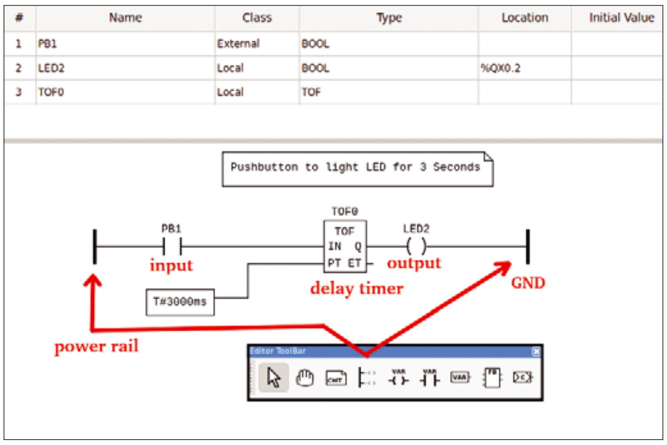

AND logic: push button 1 AND 2 to light LED
button1
button2
LED1
OR logic: push button 1 OR 2 to light LED
button1
LED2
button2

Figure 5: AND/OR logic in a ladder diagram.



#	Name	Class	Type	Location	Initial Value
1	PB1	External	BOOL		
2	LED2	Local	BOOL	%QX0.2	
3	TOF0	Local	TOF		

Figure 6: Ladder logic to light an LED with a push-button.

has a monitor page to view the logic variables.

Ladder Diagrams

Ladder logic was the first IEC 61131-3 programming language, developed as a graphic representation for circuit diagrams of relay logic hardware. The term "ladder diagram" (LD) describes the appearance of the logic, which looks a little like a ladder (Figure 5), with the left side having a vertical power rail and the right side a vertical ground rail; a series of horizontal lines, or "rungs," are wiring hardware components between the rails.

Most electricians feel very comfortable with ladder logic, and it is an effective programming method for managing digital logic. If you come from a programming background, ladder logic might feel a bit strange at first.

Figure 5 is an example of AND/OR logic to light two LEDs with two pushbuttons. In this example, both buttons need to be pushed to light LED 1, whereas pushing either button lights LED 2

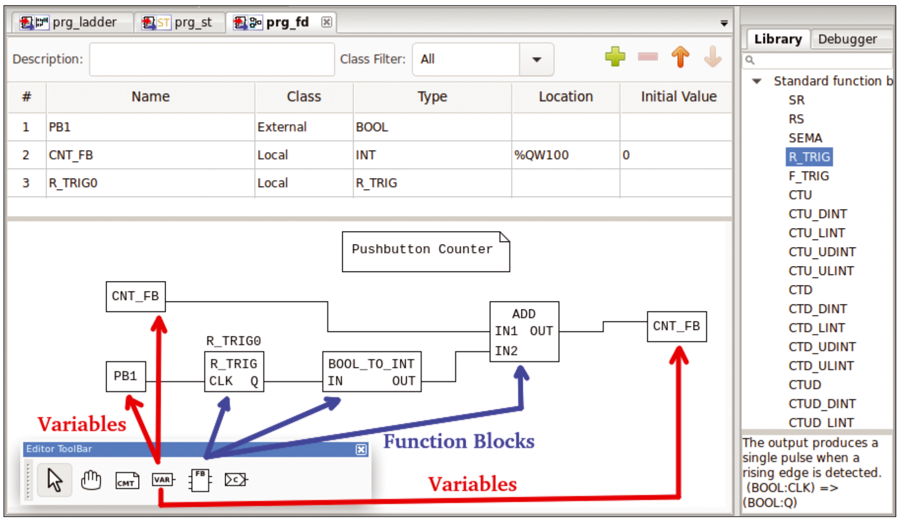
For my ladder program, I want to light an LED for three seconds with a single push of a button. In the OpenPLC editor, I referenced an external pushbutton variable, `PB1` (defined in Res0), and I created two local variables: `LED2`, my output LED, and `TOF0`, an off-delay timer.

IEC 61131-3 has a wide range of functions that can be used in ladder rungs. In this example, a timer-off delay (`TOF`) function was inserted after the pushbutton, and the time parameter (3,000ms) is wired in as a variable (Figure 6).

Function Block Diagrams

One limitation of ladder logic is that managing analog logic can be a little messy; therefore, function block diagrams (FBDs) were developed. If you are comfortable with graphic programming applications like Node-RED, you shouldn't have any problems working with FBDs.

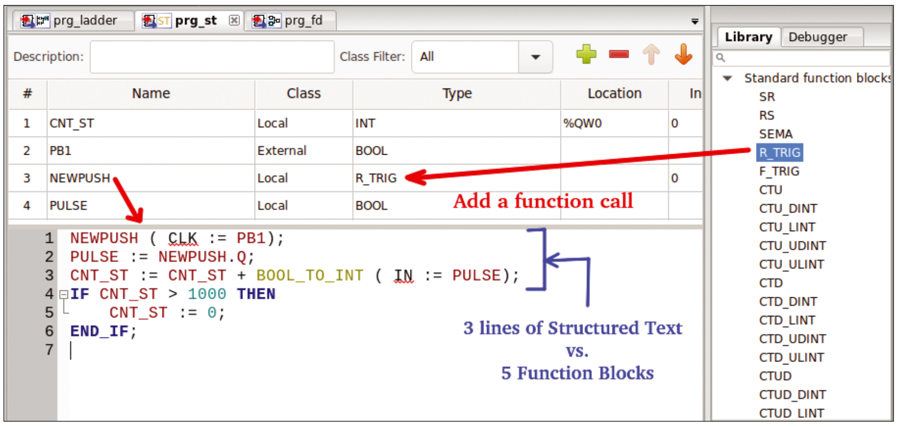
For my FBD program (Figure 7) I wanted to count the number of times the LED was lit and output the value to a Modbus holding register. As in the ladder program, the external `PB1` variable is referenced. A new output, `CNT_FB`, is defined as `%QW100`, an output word that can be accessed by the Modbus protocol at address 100. (I'll explain more about



#	Name	Class	Type	Location	Initial Value
1	PB1	External	BOOL		
2	CNT_FB	Local	INT	%QW100	0
3	R_TRIG0	Local	R_TRIG		

The output produces a single pulse when a rising edge is detected. (BOOL:CLK) => (BOOL:Q)

Figure 7: Function blocks to count pushbutton presses.



#	Name	Class	Type	Location	In
1	CNT_ST	Local	INT	%QW0	0
2	PB1	External	BOOL		
3	NEWPUSH	Local	R_TRIG		0
4	PULSE	Local	BOOL		

```
NEWPUSH ( CLK := PB1);
PULSE := NEWPUSH.Q;
CNT_ST := CNT_ST + BOOL_TO_INT ( IN := PULSE);
IF CNT_ST > 1000 THEN
    CNT_ST := 0;
END_IF;
```

Figure 8: Structured text to count pushbutton presses.

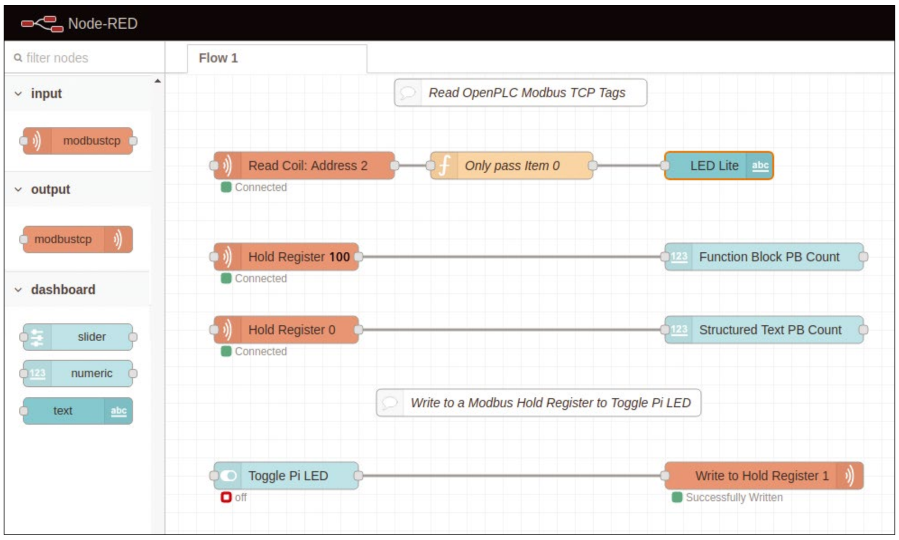
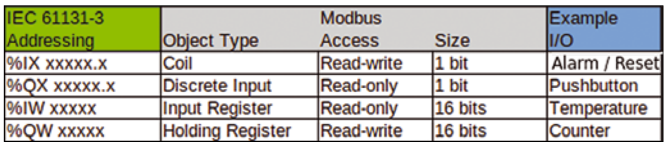


Figure 9: Modbus interface logic in Node-RED.



IEC 61131-3 Addressing	Object Type	Modbus Access	Size	Example I/O
%IX xxxxx.x	Coil	Read-write	1 bit	Alarm / Reset
%QX xxxxx.x	Discrete Input	Read-only	1 bit	Pushbutton
%IW xxxxx	Input Register	Read-only	16 bits	Temperature
%QW xxxxx	Holding Register	Read-write	16 bits	Counter

Figure 10: Modbus object types.

Modbus when I connect Node-RED to OpenPLC.)

The FBD uses a rising edge trigger (`R_TRIG`) to catch when the LED turns on. The output from `R_TRIG` is a boolean, so the value is converted to an `INT` and added to the value of `CNT_FB`.

Structured Text

One of the advantages of FBDs is that they are very readable and somewhat self-documenting. The downside of FBDs is that they can be messy for complex conditional logic. Structured text (ST) was developed as a programming option that can work along with the other 61131-3 languages. Structured text is block-structured and resembles Pascal syntactically.

For my ST program, I wanted to create the same functionality as in the earlier FBD. Interestingly, the same functionality in ST took only three lines of code (Figure 8), compared with five lines in FBD. In my ST program I also added a simple `IF` condition to reset the pushbutton counter if the value reaches *1000*.

Note that library functions such as `R_TRIG` are available in all the 61131-3 programming languages, and you can create your own custom functions in one programming language that can be used in all the other languages.

Running OpenPLC Programs

To start the runtime application manually on the Raspberry Pi, enter:

```
$ cd OpenPLC_v3
~/OpenPLC_v3$ ⮐
  sudo ./start_openplc.sh &
```

The OpenPLC runtime starts the web application on port 8080 on the Raspberry Pi.

After logging in to the web interface, the first step is to select the *Hardware* option and set the OpenPLC hardware layer to *Raspberry Pi*. Next, select the *Programs* option and upload the OpenPLC configuration file. After a new configuration file is uploaded and compiled, the final step is to press the *Start PLC* button.

For my PLC application, a button push lights the LED for 3 seconds and the function block and structured text counter variables increment up.

The *Monitoring* option can be used to view the status of variables in the PLC configuration. At this point, the PLC is working somewhat "headless," so adding a Node-RED visual interface is next.

Modbus with Node-RED

Modbus [5] was the earliest and most common communication protocol used to connect industrial devices together. Modbus can be used with serial interfaces (Modbus RTU) or on Ethernet networks (Modbus TCP); both are supported by OpenPLC.

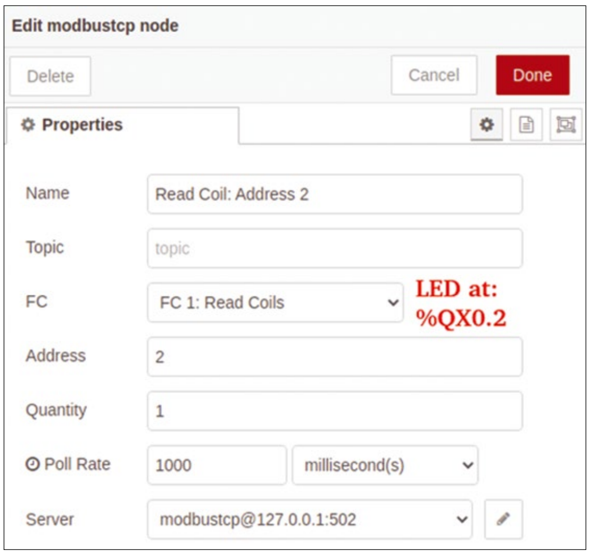
Node-RED has a number of Modbus TCP nodes that can be used. I found that `node-red-contrib-modbustcp` worked well for my application. New nodes can be added to Node-RED from the *Manage Palette* option.

A simple Node-RED application that can monitor the LED and counter statuses would use three `modbustcp input` nodes, a `text` node, and two `numeric` dashboard nodes (Figure 9).

Modbus supports four object types: coils, discrete inputs, input registers, and holding registers (Figure 10). On this Node-RED application, I am only using two types of Modbus objects: a coil (a single-digit bit, LED) and holding registers (16 bits, counters). When Modbus reads a holding register, it returns just a single value (more, if requested); however, for a coil, Modbus returns 16 bits of information, not just the single bit of interest.

To show just the LED status on a Node-RED dashboard, a small function is needed (*Only pass item 0*) to change the message payload to just the first item in the array:

```
msg.payload = msg.payload[0];
return msg;
```



Name: Read Coil: Address 2
Topic: topic
FC: FC 1: Read Coils
Address: 2
Quantity: 1
Poll Rate: 1000 millisecond(s)
Server: modbustcp@127.0.0.1:502

Figure 11: Node-RED Modbus input configuration for the Pi LED.

The `modbustcp input` node needs to be configured with the variable's Modbus object type and address. For example , the LED's IEC addressing is `%QX0.2`, which would be a coil at address 2 (Figure 11). The function block counter (`CNT_FB`) address, `%QW100`, is a holding register at address 100 (`CNT_ST` is a holding register at address 0).

Modbus Writes from Node-RED

I modified the earlier ladder logic program to light the LED from either the pushbutton or a holding register (`%QW1`; Figure 12), which is an integer, so the value is converted to a boolean then OR'd with the pushbutton interface. The result of this OR is the value of the LED.

On Node-RED, a `slider` node is used to pass a 0 or 1 to a `modbus tcp output` node, which is configured as a single write to holding register 1. After the Node-RED logic is deployed, the web dashboard is accessed at: *http://<your_rasp_pi>:1880/ui/* (Figure 13).

Final Comments

Learning industrial control theory can be a little challenging: It's a very large topic with specific standards (e.g., IEC 61131-3 programming) and industry-specific communications packages like Modbus. Luckily, open source packages like OpenPLC allow you to familiarize yourself with industrial controls on low-cost hardware like the Raspberry Pi.

OpenPLC is an excellent testing and teaching tool, but it's important to point out that OpenPLC is not designed to be used on real-time projects that have environmental or safety concerns. ■■■

Info

[1] OpenPLC documentation: *https://www.openplcproject.com/*

[2] IEC 61131-3 documentation: *https://en.wikipedia.org/wiki/IEC_61131-3*

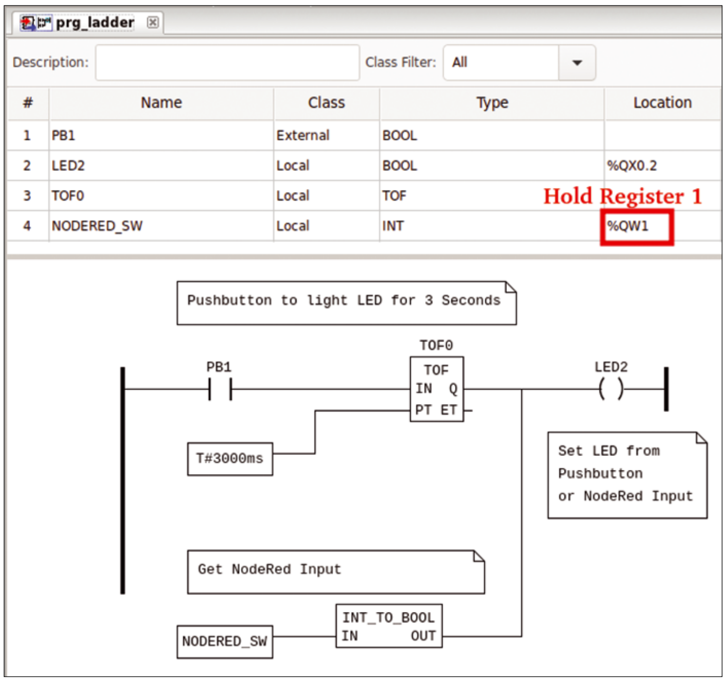
[3] Node-RED documentation: *https://nodered.org/*

[4] OpenPLC install: *https://www.openplcproject.com/getting-started/*

[5] Modbus documentation: *https://modbus.org/*

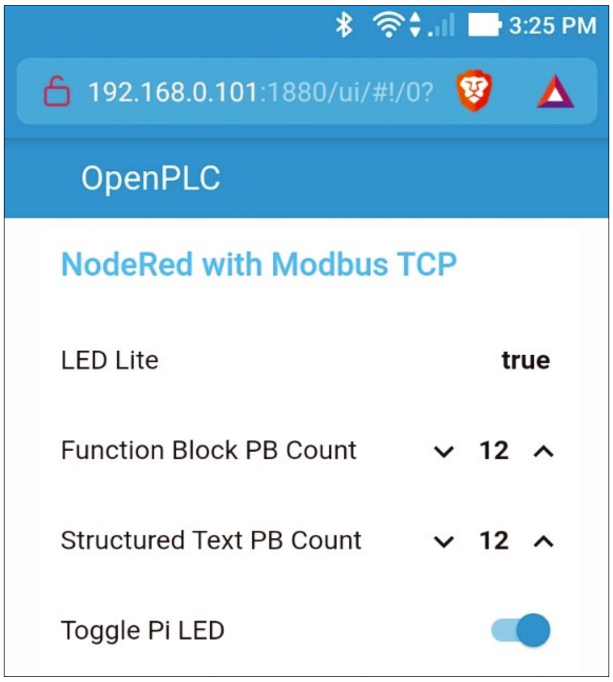
Author

You can investigate more neat projects by Pete Metcalfe and his daughters at *https://funprojects.blog*.



#	Name	Class	Type	Location
1	PB1	External	BOOL	
2	LED2	Local	BOOL	%QX0.2
3	TOF0	Local	TOF	
4	NODERED_SW	Local	INT	%QW1

Figure 12: Ladder logic to light LED from either a pushbutton or Node-RED.



NodeRed with Modbus TCP

LED Lite	true
Function Block PB Count	12
Structured Text PB Count	12
Toggle Pi LED	

Figure 13: Node-RED dashboard with read/write access to OpenPLC variables.

MakerSpace

Programmable light stick with the Raspberry Pi Pico

Painting with Light

In the photographic method of light painting, you expose a subject over an extended period of time while moving the light sources. With a little technical support from a Raspberry Pi Pico, you can achieve sophisticated results. *By Swen Hopfe*

Without light, nothing can be seen in photography, but with just a little background light and light brushes, you can create interesting motifs. For long exposures, you need a camera with manual adjustment and a tripod. A flashlight or laser pointer is sufficient to paint simple patterns or lettering into an on-going exposure. If you like to experiment, you might want to work with LEDs to achieve particularly interesting light graphics.

I was interested in a creative approach and in trying out what can be achieved with modern pixel elements and clever programming. The idea of a light stick for light painting was born. In this article, I use a Raspberry Pi Pico to build a MicroPython programmable LED light stick.

The control does not require much in the line of interfaces, and the hardware needs to be as compact as possible – a good fit for the Pico, which is the youngest member in the Raspberry Pi family and has already seen a huge amount of interest in the last months, with some interesting HATs to match. The large community proves to be a boon if you are

Table 1: Materials

Quantity	Item
1	Raspberry Pi Pico
2	LED tape WS2812b
1 each	Resistors: 460 ohm, 1.2k
1 each	Capacitor, 22nF; electrolytic capacitor, 100µF
1 each	Switch, button
2	20-pin female connector strip
1	Universal printed circuit board
4	1.2V NiMH batteries (or a 5V battery pack)
2	Aluminum U-section
	Wiring and mounting material

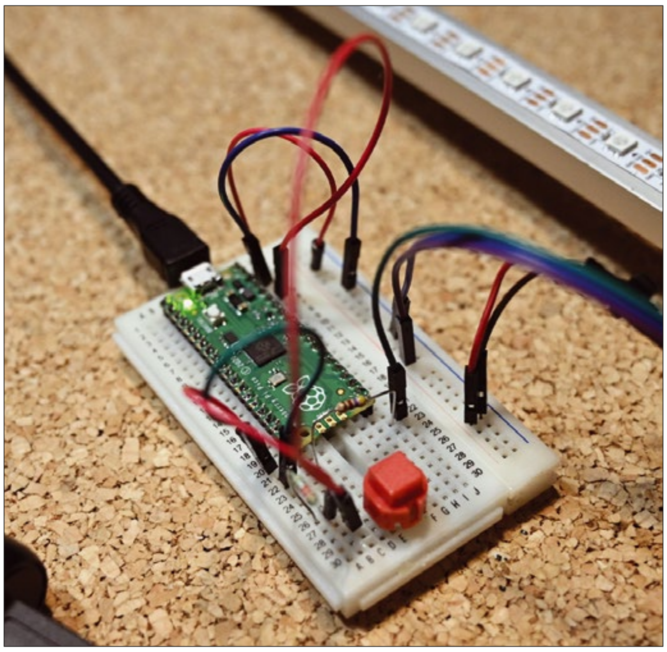


Figure 1: The circuit built on a breadboard for the initial tests.

Photo by Xuan Nguyen on Unsplash

looking for help. Currently the smallest Raspberry Pi model, the Pico comes with an RP2040 microcontroller chip and an ARM Cortex M0+ processor, offering a state-of-the-art alternative to comparable controllers. It can run C/C++ and performs well with MicroPython.

Light Brush

In this project, I use the MicroPython firmware on the Pi Pico (Table 1). The light stick does not need a WiFi connection. For a project intended for outdoor use, it is convenient to be able to switch off the supply voltage without risking damage to an SD card. I will not be going into the details of how to get started with the Pi Pico, what you need to consider when communicating with the device, and how to program with MicroPython, because enough good articles on these subjects already exist on the web.

Even a small breadboard offers sufficient space to connect the essential components for some initial tests (Figure 1). The Pico has to control a multitude of RGB LEDs, and the easiest way to implement this is with NeoPixels [1] (e.g., WS2812b), LEDs that have their own control chip and can therefore be addressed individually on the wire. Now available in many designs, they will be referred to in the following text as LEDs for the sake of simplicity.

A light stick length of 2m is fine, so it can be easily transported and, above all, moved. This movement and the timed switching of certain pixels should create a satisfactory curtain of light in the finished photo (Figure 2).

Structure

If the circuit works in the test setup, you can move on to the build. All the hardware is attached directly to the stick, which is created from a standard aluminum U-section. Luckily, the finished board turns out to be relatively compact, because with just a few components, you can work with a universal PCB (Figure 3). Ideally, the Pico will have a slot on the top side so that you can remove the PCB from the rest of the hardware. The Pi Pico is also available as an M version with soldered tips, but only a few pins have to be connected, so you can do the soldering yourself fairly quickly. That's why the board in the figure does not have a continuous socket strip.

Because I wanted to set the light points relatively close together, I decided on a strip with 60 LEDs/m, which results in a total of 120 LEDs. The LEDs require an appropriate power supply, which cannot be taken from the Pico but must be wired directly from the source. With a circuit like this, a small serial resistor in the data line to the LED strip (at GPIO18) is a good idea. Four 1.2V NiMH batteries (micro, AAA) provide power; they can be removed and charged as needed (Figure 4). Alternatively, you could use a power bank (as small as possible) that delivers

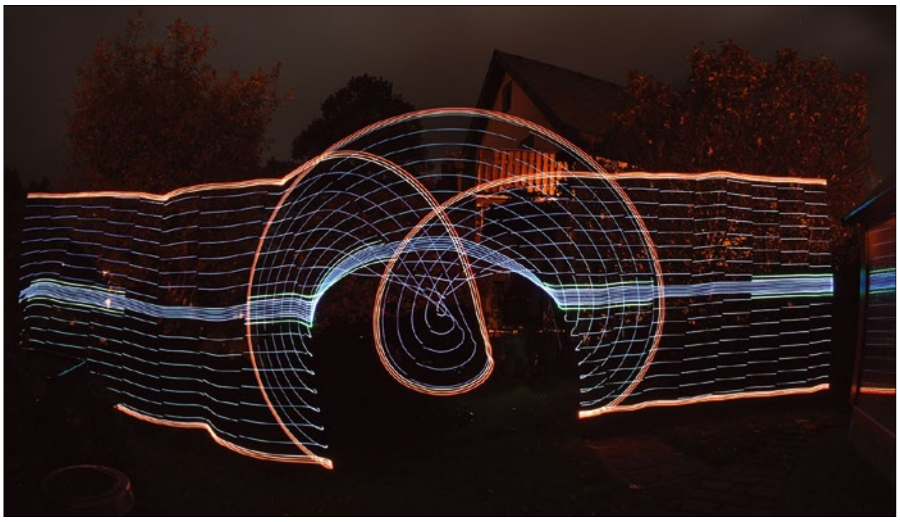


Figure 2: Painting in the air: light painting with the LED light stick and Pi Pico.

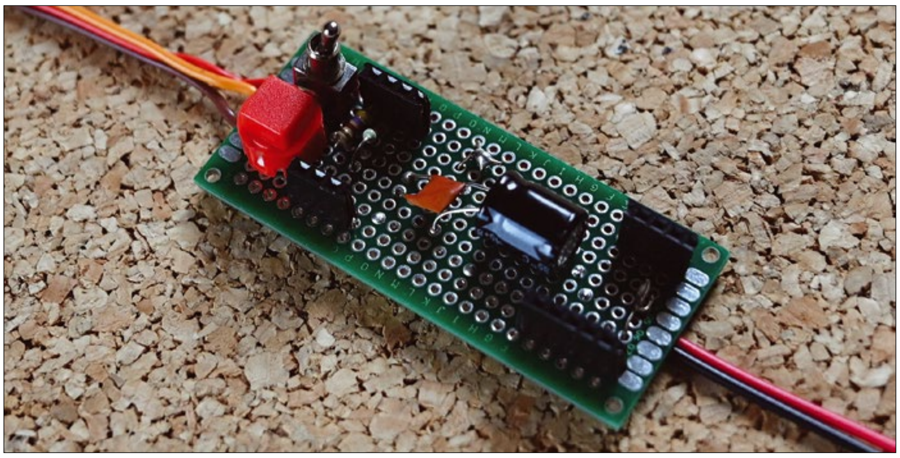


Figure 3: On the left side of the control board is an on-off switch and a push button.

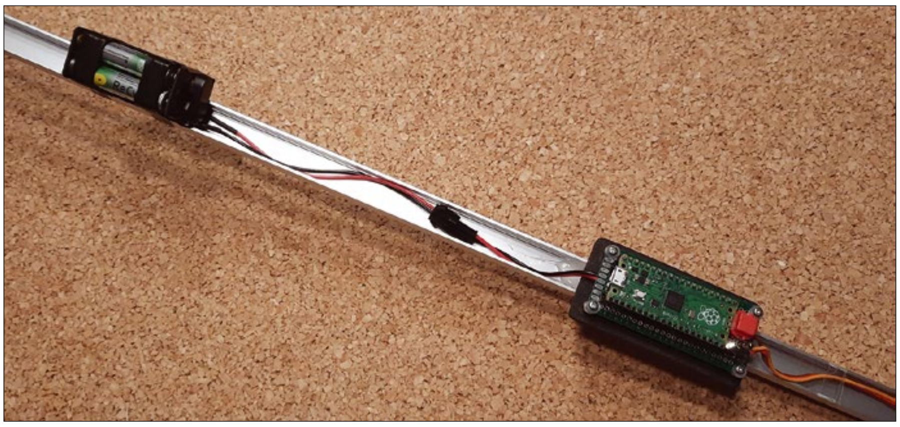


Figure 4: The control board and the batteries are positioned on the back of the light stick.

5V. At 4.8V, the circuit still works perfectly: The LEDs reach the desired brightness and can be dimmed.

Sometimes long LED arrays will manifest errors, resulting in uncontrolled flashing of the diodes that requires stabilizing the data line and power supply. This problem did not occur with the components I used. The two capacitors in the circuit diagram (Figure 5) are provided as a precaution during alternative operation with a plug-in power supply.

Operation

In the schematic in Figure 5, you can see that T1 and R2 are connected to the GPIO17 pin of the controller. When the light stick is on, processing starts first with the pre-configured light pattern. The push button is the only control element during operation, which is necessary because the camera has to be triggered before the glow stick is used, so everything must be well timed and easy to operate. Later, I will push a button to switch to the next light pattern. The green LED on the board points backward toward the operator when shooting to provide orientation for the various flash codes.

After applying the supply voltage with a toggle switch (S1), the board LED flashes every second under software control, indicating that everything is ready. At the first button press, the flash frequency drops to two seconds, and you now have only a short time to get into recording position. Immediately before the start of the light sequence, the diode flashes very briefly, at which time you have to start moving the light stick. Everyone can create their own sequences with various levels of operating convenience.

The Pico has a control button onboard for switching to mass storage mode. If the microcontroller is connected to the computer over USB, a window with a file manager opens on Linux or Windows, and you can drag and drop source code there. A C/C++ development environment is built in, and Arduino IDE integration works, as well.

The easiest tool for Python development is Thonny [2]. Once the lean IDE (Figure 6) has been set up for the Pi Pico and MicroPython interpreter, communication with the module is automatic. The pleasingly clear-cut editor can be configured in all of its basic features. In the Save dialog, you can choose whether to upload the source code directly to the Pico or store it in a directory, which proves to be particularly useful in practice.

The real creativity in this project lies in the design of the sequences for the light control. The upright light stick maps the lines of the light image virtually as it moves, and you have to consider which LEDs you want to light up and at what intervals, color, and intensity so that they end up painting an interesting picture (Figure 7).

The interesting thing for the developer is that the PIO assembly and up to

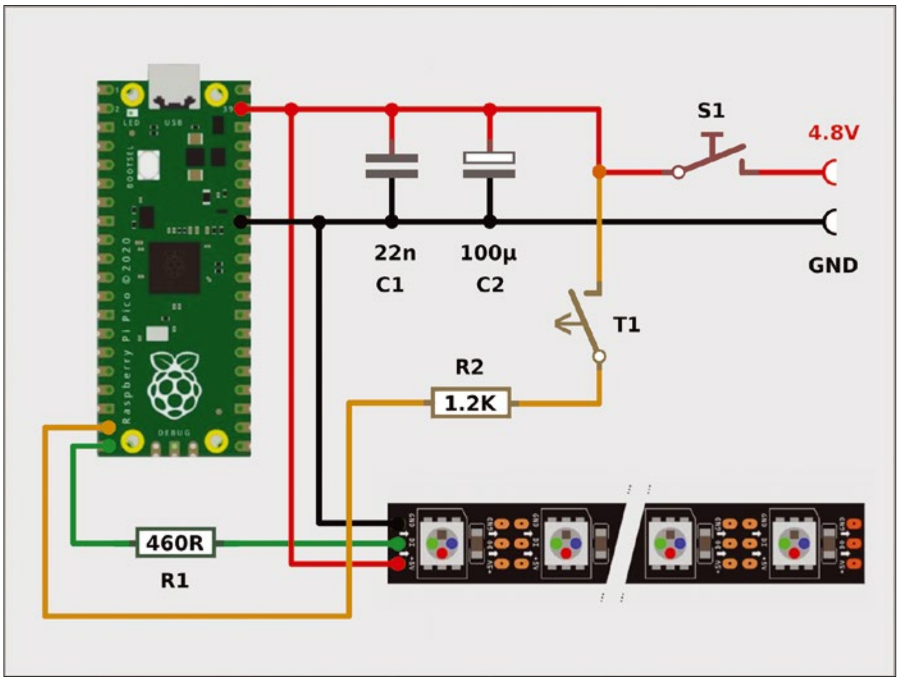
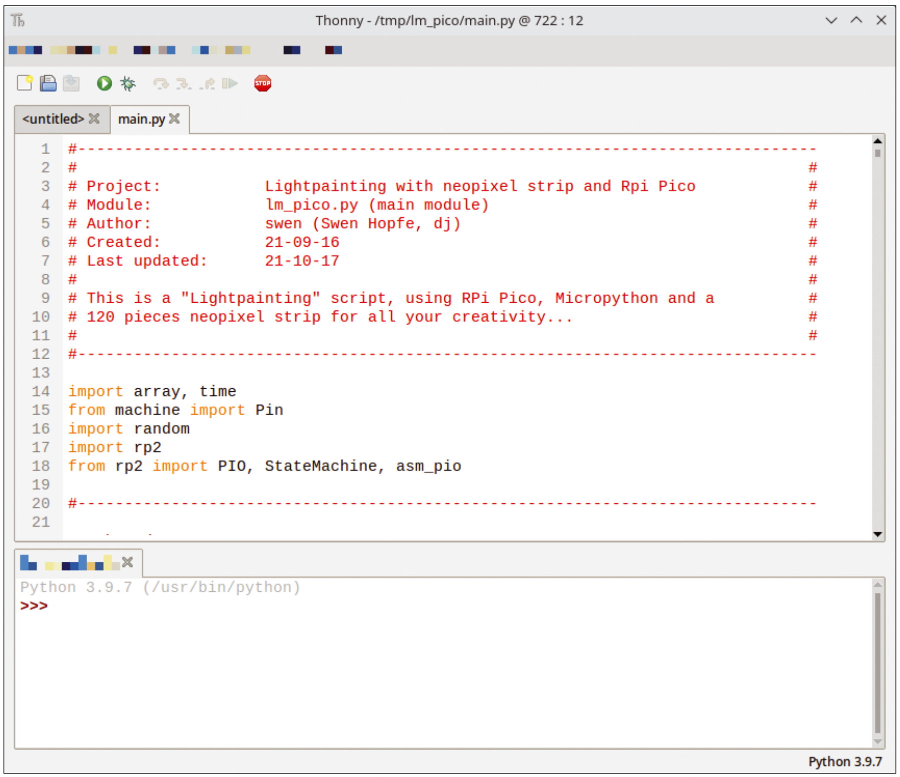


Figure 5: The button (T1) in the circuit diagram is the only control element for the light bar.



```
#-------------------------------------------------------------------------------
#                                                                              #
# Project:          Lightpainting with neopixel strip and Rpi Pico             #
# Module:           lm_pico.py (main module)                                   #
# Author:           swen (Swen Hopfe, dj)                                      #
# Created:          21-09-16                                                   #
# Last updated:     21-10-17                                                   #
#                                                                              #
# This is a "Lightpainting" script, using RPi Pico, Micropython and a          #
# 120 pieces neopixel strip for all your creativity...                         #
#                                                                              #
#-------------------------------------------------------------------------------

import array, time
from machine import Pin
import random
import rp2
from rp2 import PIO, StateMachine, asm_pio

#-------------------------------------------------------------------------------
```

```
Python 3.9.7 (/usr/bin/python)
>>>
```

Figure 6: The simple, well-structured Thonny development environment with the source code for the project.

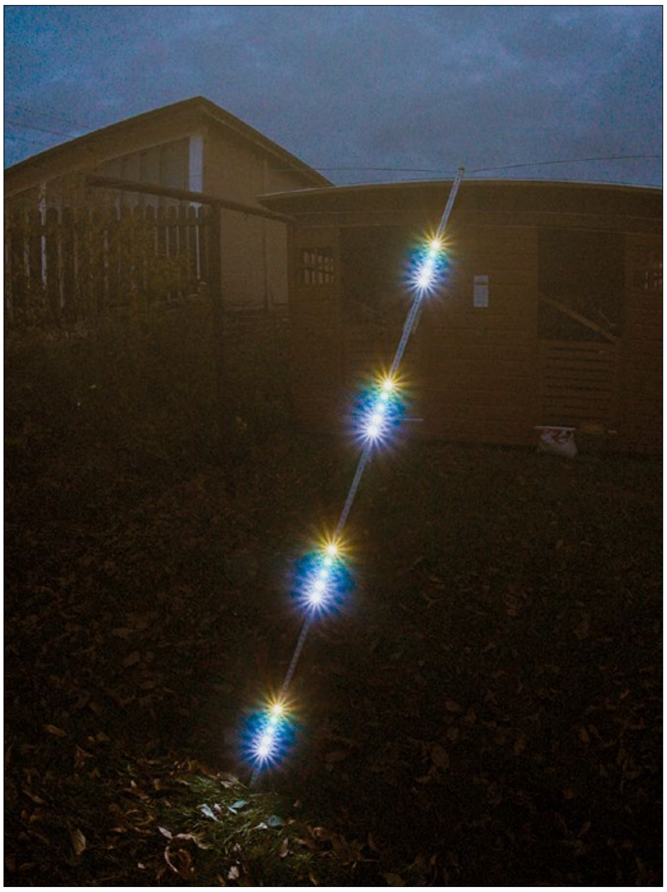


Figure 7: All 120 LEDs of the light stick can be controlled independently.

eight state machines can be defined in MicroPython on the Pico, so all data can be transferred fast enough to the single-wire bus that controls the LED pixels. It is a good idea to follow the documentation first and seek guidance in blogs and wikis.

After a definition block, the next part continues with some functions that update the colors of all or selected pixels – optionally in combination with a list of brightness values as used by the light control. The project source code, which can be found on my GitHub page [3], contains some sample methods, as well as a fair amount of information about the rest of the programming needed.

Camera Settings

For an initial shot, I went outdoors in the evening, taking care to wear dark clothing. The camera was set up on a tripod a few meters away. With a short focal length, you can capture as much of the surroundings as possible. To have sufficient time for the shot, I set the exposure time to 20 seconds at a low ISO (not greater than 800, depending on the camera). Test shots helped me determine how much of the background could be captured with a chosen aperture setting. Depending on how much ambient light you have at dusk, the aperture can be *f*/8.0 or smaller.

If the camera's automatic focus doesn't work because of the low light level, a little trick helps: Focus on a bright object nearby; if necessary, shine a flashlight on it. Once you are confident, measure the distance and adjust the focus manually. In the end, with a little improvisation, the picture can be taken without an expensive camera system.

Some interesting light sequences are prepared in the source code of the project. The animations all run for about 15 seconds, so always a bit shorter than the exposure time. When shooting, it is important to move with as steady a hand as possible. For circular shots, the light stick offers a screw connection exactly in the middle. You can change everything, with nothing to limit your creativity (Figure 8). Compared with a static illuminated object, wherein only the same traces and patterns can be seen in the same image, this project shows its strengths. As far as favorable brightness and the timed sequence of LEDs are concerned, a targeted adjustment succeeds after only a few tests.

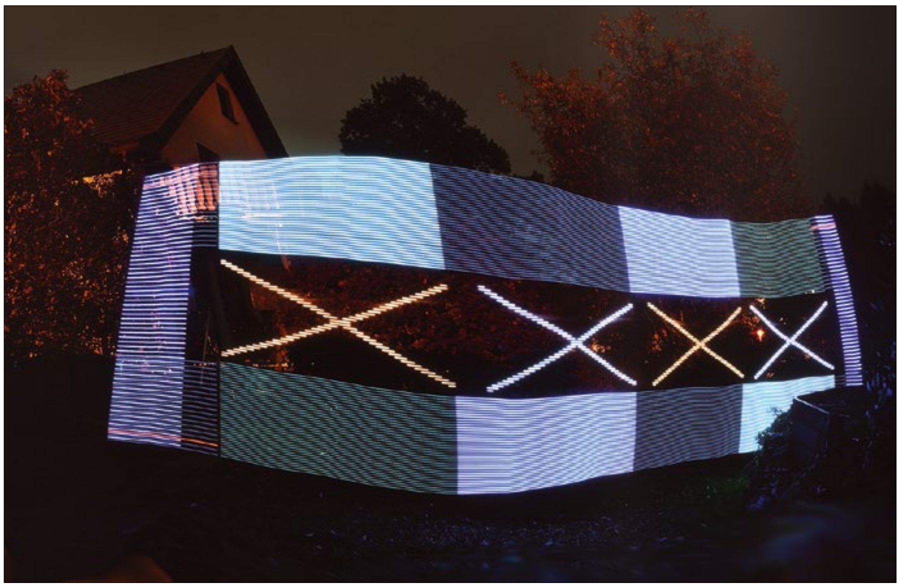


Figure 8: With a little imagination and programming skills, many motifs are possible.

Conclusions

The Raspberry Pi light stick project has certainly not made me a light painting expert. Far more spectacular shots can be found online. However, combining my own interest in the Raspberry Pi and in photography gave me a lot of pleasure, and I still have a lot of fun designing new light images. Results from the Pi Pico toolset and MicroPython are completely satisfactory. ■■■

Info

[1] Adafruit NeoPixel: *https://www.adafruit.com/category/168*

[2] Thonny: *https://thonny.org*

[3] Source code: *https://github.com/swenae/lm_pico*

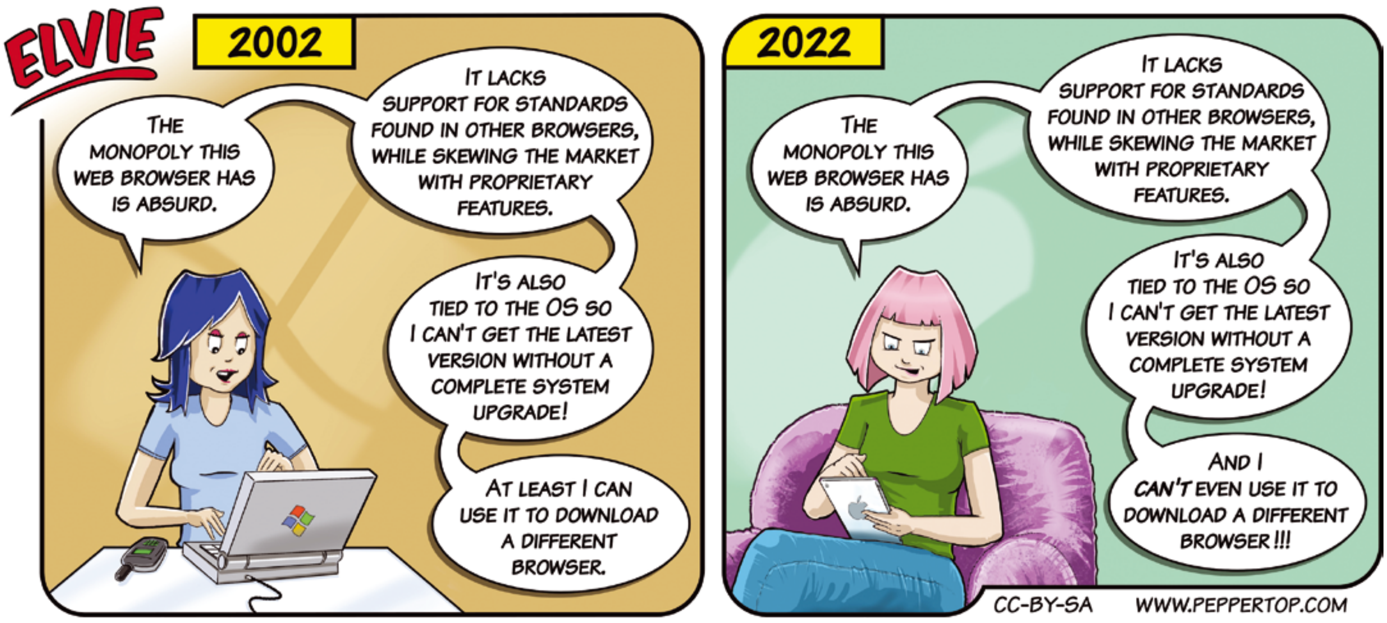
Author

Swen Hopfe works for a medium-sized company with a focus on smart cards and near-field communication (NFC). When he is not taking pictures or out and about in nature, he focuses on Raspberry Pi, IoT, and home automation projects.

One thing people love about open source is the spirit of innovation. If you can visualize a solution, you can build it yourself. Users all around the world have been expanding and extending the Linux experience for years, so at this point, you don't even have to be a coder – you can choose a solution from the great gallery of components that are available right now in the Linux universe. We serve up a pair of desktop innovations in this month's Linux Voice: Fly-Pie, a new way to think about desktop menus, and CopyQ, a souped up clipboard with some special features. Also inside: We introduce you to PhotoPrism, a cool tool that brings the power of artificial intelligence to your photo library.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
2002
THE MONOPOLY THIS WEB BROWSER HAS IS ABSURD.
IT LACKS SUPPORT FOR STANDARDS FOUND IN OTHER BROWSERS, WHILE SKEWING THE MARKET WITH PROPRIETARY FEATURES.
IT'S ALSO TIED TO THE OS SO I CAN'T GET THE LATEST VERSION WITHOUT A COMPLETE SYSTEM UPGRADE!
AT LEAST I CAN USE IT TO DOWNLOAD A DIFFERENT BROWSER.
2022
THE MONOPOLY THIS WEB BROWSER HAS IS ABSURD.
IT LACKS SUPPORT FOR STANDARDS FOUND IN OTHER BROWSERS, WHILE SKEWING THE MARKET WITH PROPRIETARY FEATURES.
IT'S ALSO TIED TO THE OS SO I CAN'T GET THE LATEST VERSION WITHOUT A COMPLETE SYSTEM UPGRADE!
AND I CAN'T EVEN USE IT TO DOWNLOAD A DIFFERENT BROWSER!!!
CC-BY-SA
WWW.PEPPERTOP.COM

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®.

MADDOG'S DOGHOUSE

Hiring the best candidates for tech jobs and increasing diversity and opportunities in the industry go hand-in-hand.

BY JON "MADDOG" HALL

Opening doors in the tech industry

Several topics have come up over the past couple of weeks, and most of them have to do with the "Technology Industry."

The first is the simplest. It is represented by a conversation that was started by a friend of mine when he said two people had come up to him in the past week and basically said, "I am tired of living in poverty; I am going to 'learn tech'."

This was a recognition that in their current line of work they were making minimum wage, and they saw that programmers, systems administrators, and others "in tech" were making a comfortable living at something which (barring mental incapacity) they might do for the rest of their lives. And they wanted that type of job, that type of life.

Of course many people reading this will realize that not all people have the skills to "do tech," but many people do develop those skills when given the same types of training and practice needed to be successful in any career. It is probably a lot more likely that a person will be able to study computer science and have a reasonable standard of living than that they might be able to rise to that level in other "glamorous" occupations.

The second item, related in a way, is diversity. Currently an organization that I am working with is recognizing that we need to work harder on diversity. To me diversity is all about economic freedom. An organization or community, barring any real factors, should match the general diversity of the population. In my experience, given the same access to education and opportunity, there should be an equal number of women as men in fields such as programming and systems administration. If there are not an equal number, then, from what I can see, there is some type of outside influence that is creating this inequality and that influence should be eliminated.

When I went to Malaysia many years ago, 70 percent of the people in programming were women. When asked why, I was told that at the age of 15 Malaysian men could get jobs doing physical labor at which the women would typically not qualify. Therefore women went on to university to study computers and management and therefore got the skills to do those jobs. The Malaysian government had an active program to bring the numbers to 50/50, to reflect the population.

When I started programming in 1969, more than 40 percent of the programmers and people in middle management of programming were women. Over time those numbers of women became less and less. It was hard to see why it was happening, and there are many theories, but over time there were fewer and fewer women in programming and systems administration, as well as a glass ceiling for women managers that seemed to mirror industry in general.

The same could be said for Black people. There were way fewer Black people in computing in the USA than in the general population.

When I taught at Hartford State Technical College (HSTC) from 1977-1980, I was proud of the programs we had to actively bring women into the technical community. One program, "Women In Technology" (WIT), was aimed at women who had a technology degree but had decided to leave industry to start and raise a family. After their children were old enough to be left at home, these women would want to re-enter the technical world, but their skills needed "refreshing." After spending nine months at HSTC in the WIT program, these women could return to industry and make their way in the workforce. Our WIT program had a 98 percent placement record and was often funded fully by grants to the women, so it was tuition free.

Many people that read my articles or blogs know that I am gay. I have been fortunate to be in an industry that, for the most part, respects LGBTQ+ people for what they do and not just what they are. However, there are still people in the LGBTQ+ community who are not welcome in every company, such as transgender people. For some reason that I really can not figure out, transgender people receive that "special" type of hatred, almost as if people think that if transgender people did not try to transition they are somehow "less" transgender. I have news for you … they are still just as transgender, even if you do not know it.

So I am particularly proud that some of my best friends are helping transgender people get jobs in technology, raising their income levels and contributing to society.

And so it goes. I do not want the richest, the whitest, the most macho people in the tech field. I simply want the best. I hope you do too. ■■■

Fly-Pie navigates through the system with pie menus

Piece of Pie

No matter whether you use Gnome or KDE, Windows or macOS, menus always pop up from a bar. Fly-Pie organizes a freely configurable menu in the form of a pie chart instead. BY CHRISTOPH LANGNER

The way we control a desktop environment has not changed significantly since the early days of Gnome, KDE, and Xfce, even if Gnome in particular tends to take a swipe at the paradigms every now and then. Even early interface role models like GEM or the Xerox Alto operating system, the forefather of all graphical user interfaces, used graphical elements such as windows, scrollbars, or menus. Every now and then, however, it is useful to step off the familiar path and try something new. The Fly-Pie [1] extension for the Gnome desktop is a candidate for an off-the-beaten-path excursion.

In contrast to more heavily keyboard-oriented approaches such as the Gnome Shell's activity overview or launchers like Kupfer [2], Fly-Pie is aimed at users who prefer to keep their hands on the mouse instead of reaching for the keyboard. The dynamic pie menu lets you launch applications or move them to the foreground, simulate hotkey presses, and much more. The menu can be customized in a granular way thanks to the comprehensive configuration manager.

Control by Gesture

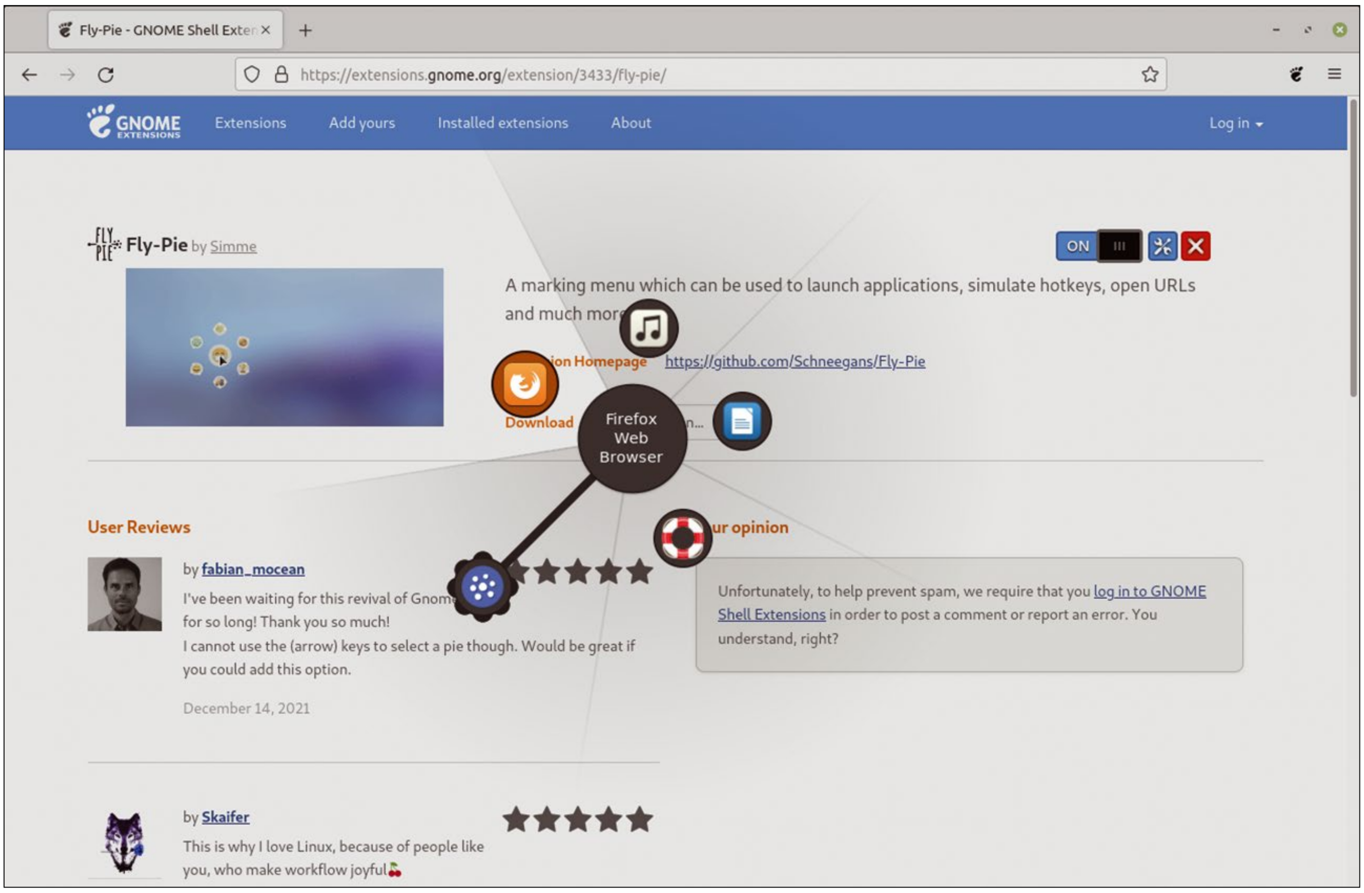
Fly-Pie is not a standalone program but an extension for the Gnome desktop version 3.34 or newer. To install it, you simply open the Gnome Shell Extensions web page [3], move the black slider from *Off* to *On*, and select *Install* (see the "Gnome Extensions" box). After that, the extension appears in the Gnome extension manager as *manually installed*. From there, you can also access the settings via the gear icon. I tested the extension on Arch Linux with the current Gnome 41.

For an initial test, press Ctrl+Space: This pops up a menu in the form of a pie chart at the mouse pointer position. Depending on the sector you choose, you can trigger different actions by clicking. It does not matter how far away from the center you are when you perform the action. In the preconfigured example menu, you can use the icons in the lower half to jump to the next or previous desktop, close or maximize the current window, or open the Fly-Pie settings.

The top sectors *Running Applications*, *Sound*, and *Favorites* branch to a new pie menu, as indicated by the "bubbles" around the sector's icon. For example, to open the Firefox browser, press Ctrl+Space, then tap the heart icon and the Firefox icon in the submenu. Alternatively, you can use highlight mode. In this case, click with the mouse pointer on an entry in the Fly-Pie menu and hold down the left mouse button.

Gnome Extensions

To install Gnome extensions via the website (*https://extensions.gnome.org*), you need a compatible web browser with a special browser extension. Extensions are available for Chrome [4], Opera, and Firefox [5] but not for the in-house Gnome browser Web, which itself does not yet support extensions. In addition, the *chrome-gnome-shell* package from the package manager must be installed. This is not part of the default installation on every distribution. Don't be confused by the name, you still need the package in combination with Firefox. Unfortunately Ubuntu, with its own special way of installing browsers such as Firefox and Chromium as Snap packages, causes issues here. The browser fails to communicate with Gnome Shell. As a workaround, you could always install the Chrome browser.



Fly-Pie by Simme

A marking menu which can be used to launch applications, simulate hotkeys, open URLs and much mor

https://github.com/Schneegans/Fly-Pie

User Reviews

by fabian_mocean

I've been waiting for this revival of Gnom for so long! Thank you so much!
I cannot use the (arrow) keys to select a pie though. Would be great if you could add this option.

December 14, 2021

Unfortunately, to help prevent spam, we require that you log in to GNOME Shell Extensions in order to post a comment or report an error. You understand, right?

by Skaifer

This is why I love Linux, because of people like you, who make workflow joyful🍒

Figure 1: The example Fly-Pie menu in action: You can move the menu to the foreground by pressing Ctrl+Space.

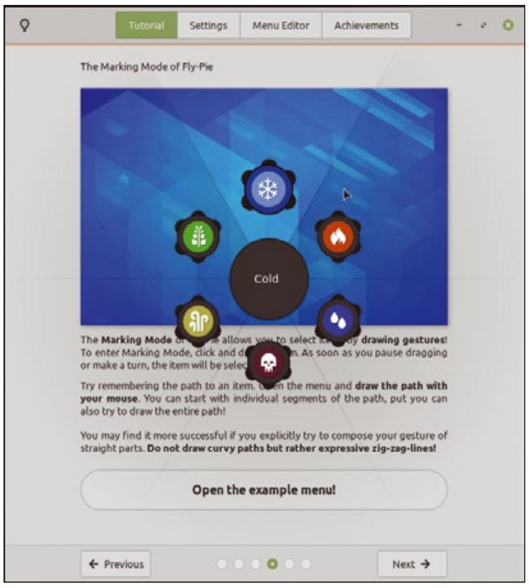
Then drag the mouse pointer a little away from the center and stop the movement. At this point, the next submenu will open automatically, if available, or the corresponding action will be performed.

The first two modes are basically just for learning how things work. Fly-Pie is more efficient in Turbo mode, which means significantly less mouse clicking. To do this, call up the Fly-Pie menu again by pressing Ctrl+Space. Then click on the center and hold down the mouse button. To open the Firefox browser, drag the mouse pointer over the heart icon. Briefly hover over the heart icon, wait until the menu opens, and continue to the Firefox icon. When you get there, release the left mouse button again (Figure 1).

Practical Exercises to Get Started

The principle is not easy to understand at the beginning, which is why Fly-Pie offers users a detailed introduction. You can access this in the program settings. Open the menu by pressing Ctrl+Space and then click on the gear icon at the bottom. The introduction guides you in several steps through the various operating modes. Practical examples and videos explain how Fly-Pie works (Figure 2). At the end, there are even awards to be won once you can control the program quickly enough.

However, Fly-Pie shows its real strengths when you start creating your own menus or adapt the sample menu to your needs. To do this, open *Menu Editor* on the *Settings* tab. Click on the pencil icon on the right above the star to enable editing mode (Figure 3). Now select the menu. You can change the name and icon and even define a new keyboard shortcut to open it. To delete, drag a



The Marking Mode of Fly-Pie

The **Marking Mode** o e allows you to select by **drawing gestures**! To enter Marking Mode, click and d n. As soon as you pause dragging or make a turn, the item will be selec

Try remembering the path to an item. en the menu and **draw the path with your mouse**. You can start with individual segments of the path, put you can also try to draw the entire path!

You may find it more successful if you explicitly try to compose your gesture of straight parts. **Do not draw curvy paths but rather expressive zig-zag-lines!**

Figure 2: The Fly-Pie introduction explains step-by-step how the dynamic pie menus work.

menu's icon to the *Trash* area at the bottom of the screen. The *Stash* area next to it temporarily disables a menu without immediately and permanently deleting it.

To create your own menu, click the plus icon to the right of the All Menus menu item below the titlebar. The dialog shows a number of predefined entries such as *Devices*, *Bookmarks*, and *System*, all of which already contain predefined menus or actions. Optionally, start on an empty page with a *User Defined Menu*. Starting from the basic menu, you can then add further submenus by pressing the plus icon. You can freely organize the individual entries in each sector. You may want to arrange important actions in a line so that you only need to move the mouse pointer straight through the menu structure.

Conclusions

Fly-Pie's greatest strength lies in its flexibility. The program does not limit itself to a single menu but lets you create separate menus for important applications or tasks, each of which can then be called up with its own keyboard shortcut. For example, you can create a menu for operating the media player, one for navigating the web, and one for word processing with your favorite editors, including links to the most important folders and sources.

The project is a fine example of a Gnome extension. The introduction helps users understand operations. Videos explain the most important actions step-by-step, and the developer presents the latest innovations on the project page. The small game with its awards also motivates people to use the program as efficiently as they can to discover the best possible approach to using Fly-Pie. ∎∎∎

Info

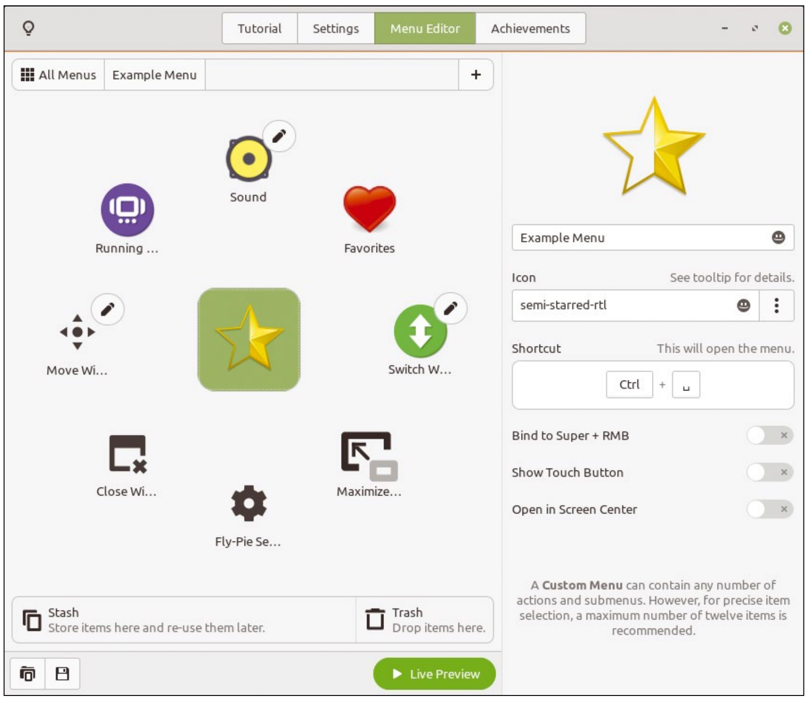
[1] Fly-Pie: *http://schneegans.github.io/*

[2] Kupfer: *https://kupferlauncher.github.io*

[3] Gnome Shell extensions: *https://extensions.gnome.org/extension/3433/fly-pie/*

[4] Gnome Shell integration for Chrome: *https://chrome.google.com/webstore/detail/gnome-shell-integration/gphhapmejobijbbhgpjhcjognlahblep*

[5] Gnome Shell integration for Firefox: *https://addons.mozilla.org/en-US/firefox/addon/gnome-shell-integration*



Example Menu

Icon — See tooltip for details.

semi-starred-rtl

Shortcut — This will open the menu.

Ctrl + ␣

Bind to Super + RMB

Show Touch Button

Open in Screen Center

A **Custom Menu** can contain any number of actions and submenus. However, for precise item selection, a maximum number of twelve items is recommended.

Figure 3: ***Menu Editor*** **lets you adapt the example to suit your own needs or create completely new menus from scratch.**

An advanced clipboard manager

Clipboard Deluxe

CopyQ extends the clipboard with practical everyday functions while also catering to advanced needs. BY FERDINAND THOMMES

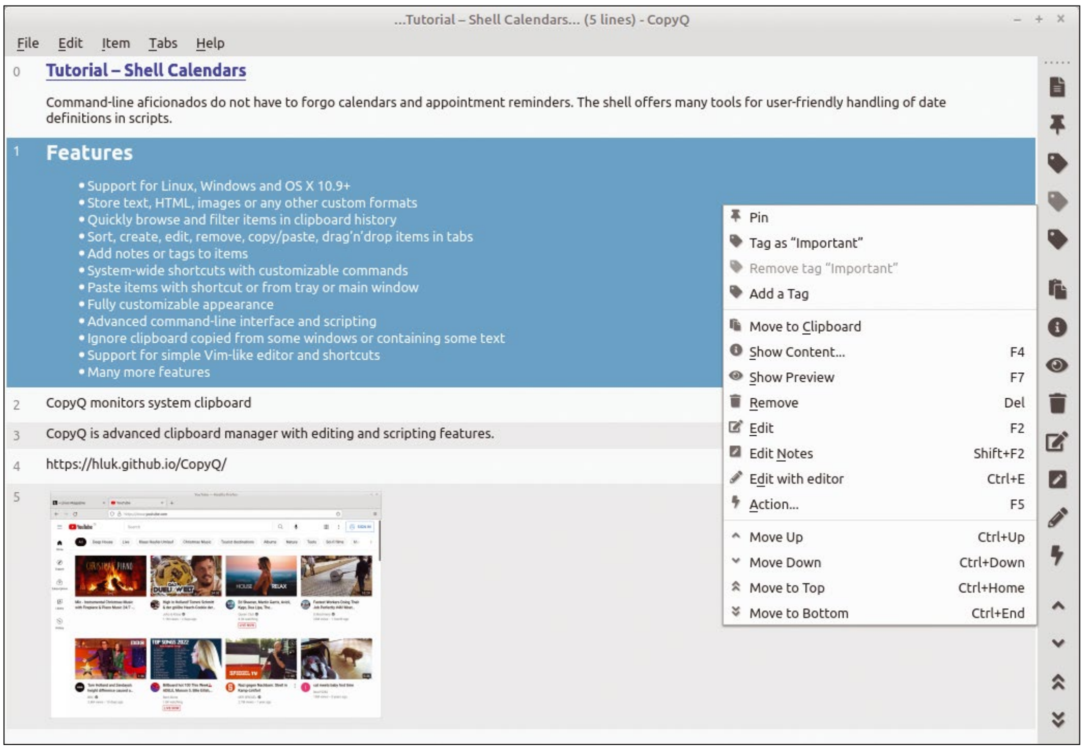
Regardless of whether you use Linux, macOS, or Windows, the clipboard is one of the most frequently used tools on the desktop. It is a buffer provided by the operating system that temporarily stores, for instance, text snippets which can then be copied and pasted into other applications.

The various desktop environments often have clipboard managers that extend the possibilities of the clipboard. In contrast to the operating system's own clipboard, which resides in RAM and whose contents thus disappear at the next reboot, these small tools help preserve a configurable number of texts beyond a reboot. In KDE Plasma, the corresponding application is known as Klipper, which is loosely derived from the clipboard.

You populate the clipboard by pressing Ctrl+X, Ctrl+C, or by selecting something with the mouse and calling the corresponding item from the context menu. Applications such as screenshot tools or password managers also have a menu item for saving recordings or passwords in Klipper.

In the settings, you specify whether Klipper starts with a blank memory for security reasons or continues to keep the previous contents across sessions. In addition, actions can be specified with saved text sections, such as automatic pasting or including regular expression-based pastebins.

Even though Klipper and its ilk already extend the clipboard's capabilities considerably, there's still far more that can be done. CopyQ [1], an



0 Tutorial – Shell Calendars

Command-line aficionados do not have to forgo calendars and appointment reminders. The shell offers many tools for user-friendly handling of date definitions in scripts.

1 Features

- Support for Linux, Windows and OS X 10.9+
- Store text, HTML, images or any other custom formats
- Quickly browse and filter items in clipboard history
- Sort, create, edit, remove, copy/paste, drag'n'drop items in tabs
- Add notes or tags to items
- System-wide shortcuts with customizable commands
- Paste items with shortcut or from tray or main window
- Fully customizable appearance
- Advanced command-line interface and scripting
- Ignore clipboard copied from some windows or containing some text
- Support for simple Vim-like editor and shortcuts
- Many more features

2 CopyQ monitors system clipboard

3 CopyQ is advanced clipboard manager with editing and scripting features.

4 https://hluk.github.io/CopyQ/

5

Figure 1: As a unique selling point, CopyQ lets you display saved images instead of just saving a reference to them.

open source clipboard management tool that was first released in 2009, offers a multi-platform solution. Furthermore, CopyQ is the only clipboard manager I know of that stores and displays images in addition to text and rich text (RTF) (Figure 1). Competitors such as Klipper or GPaste only keep references to images and cannot display them.

CopyQ is not only available for Linux but also for Windows and macOS, so you don't have to change horses just because you use different operating systems. CopyQ can be installed from the archive of most Linux distributions and is currently available in version 6.0.1. If the distribution you are using installs the suggested packages ("recommends") with the program, you just need to set up the *copyq* package; otherwise you also need *copyq-plugins*.

Long-Term Memory

After installing CopyQ, your computer's new long-term memory is immediately ready for use. It will automatically adopt all clipboard content in the future. If you want to stop this temporarily, switch off the feature in the main menu below *File* | *Disable Clipboard Storing* or by Ctrl+Shift+X. You can achieve the same effect with the commands `copyq disable` and `copyq enable` in a terminal window.

Using the basic functions turns out to be self-explanatory, while the learning curve is a little steeper for the advanced functions. After checking out the very extensive setting options (Figure 2), you will certainly appreciate some detailed documentation [2].

Regarding everyday operations. CopyQ sits in the system panel with an icon featuring a pair of scissors with green handles. Clicking on the icon opens CopyQ's main window, which you can alternatively access at the command line with the `copyq toogle` command.

Right-clicking on the icon (or using the `copyq menu` command) brings up a context menu, as expected; it displays the last five saved entries in the upper area by default and offers access to the settings below, among other things. Invoked in this way, there is also a search bar in focus at the top of the window that you type into directly. The `copyq help` command shows all commands that CopyQ supports at the command line.

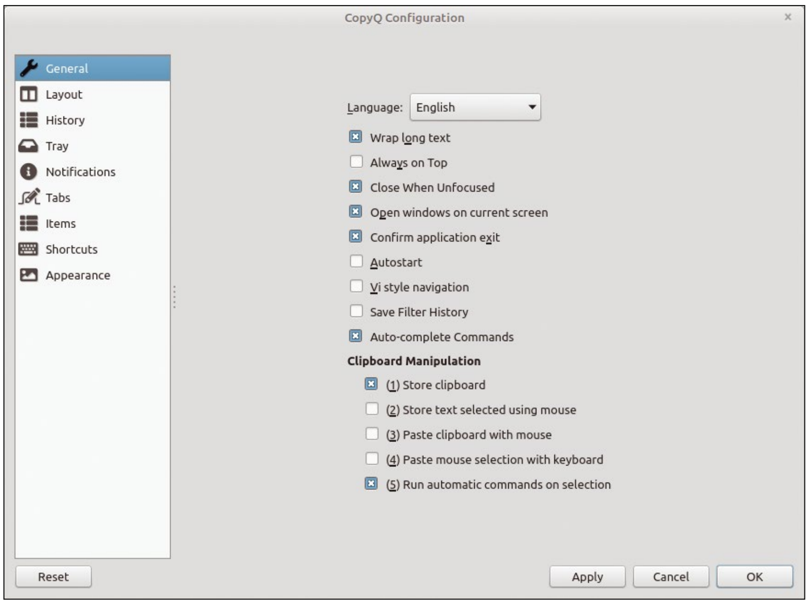
Copy & Paste

By far the most common use case in everyday life is copying and pasting content. If you need recent entries, you can do this in the main window or the context menu, which, as mentioned, displays the last five entries. This number can easily be increased to 10, for example. However, it does not make sense to use more than 10 entries for sake of a better overview. The main window is the better choice if you want more.

No matter which of the two windows you are using, after selecting the desired entry, a double-click beams it to the cursor position in the respective application. To do this, you first need to enable the *Paste in current window* entry in the settings below *Progress*. You can copy selected elements back to the clipboard by pressing the Enter key or Ctrl+C.

The Look

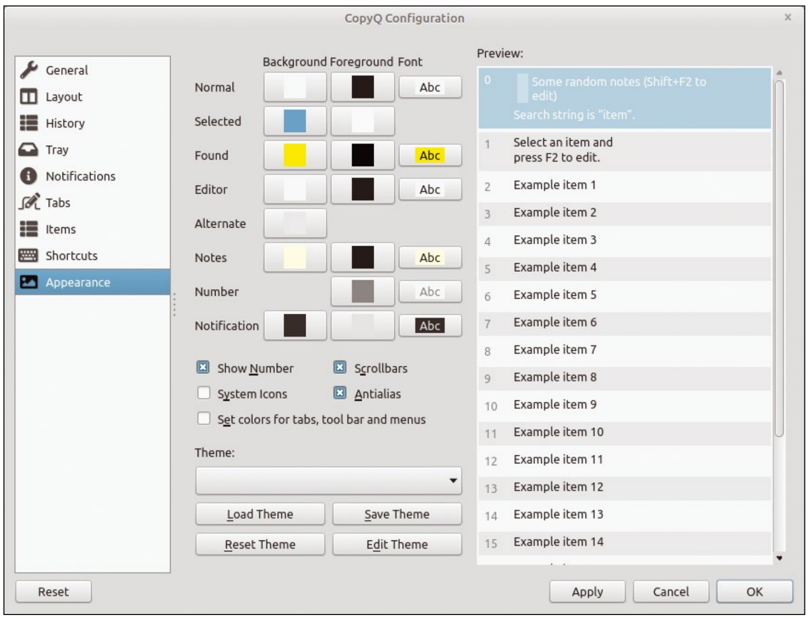
For a better overview with many stored entries you can create topic-related tabs. Under the Tabs menu, you define new tabs, rename existing ones, and delete those that are no longer needed. With the left arrow and right arrow keys you switch between tabs. For example, I use a daily tab that holds frequently used shortcuts



Language: English
- ☒ Wrap long text
- ☐ Always on Top
- ☒ Close When Unfocused
- ☒ Open windows on current screen
- ☒ Confirm application exit
- ☐ Autostart
- ☐ Vi style navigation
- ☐ Save Filter History
- ☒ Auto-complete Commands

Clipboard Manipulation
- ☒ (1) Store clipboard
- ☐ (2) Store text selected using mouse
- ☐ (3) Paste clipboard with mouse
- ☐ (4) Paste mouse selection with keyboard
- ☒ (5) Run automatic commands on selection

Figure 2: The existing default settings turn out to be self-explanatory, but some of the advanced features might require studying the detailed documentation.



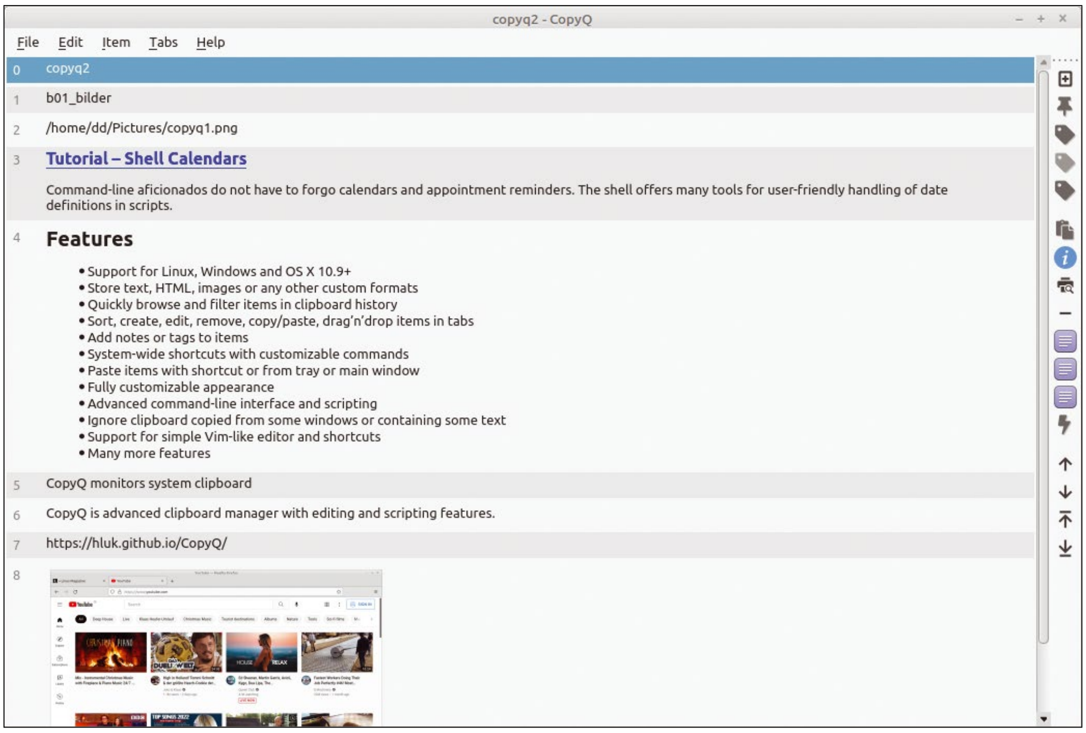
	Background	Foreground	Font
Normal	⬜	⬛	Abc
Selected	🟦	⬜	
Found	🟨	⬛	Abc
Editor	⬜	⬛	Abc
Alternate	⬜		
Notes	⬜	⬛	Abc
Number		▨	Abc
Notification	⬛	⬜	Abc

- ☒ Show Number
- ☒ Scrollbars
- ☐ System Icons
- ☒ Antialias
- ☐ Set colors for tabs, tool bar and menus

Theme:

Preview:
0 Some random notes (Shift+F2 to edit)
Search string is "item".
1 Select an item and press F2 to edit.
2 Example item 1
3 Example item 2
4 Example item 3
5 Example item 4
6 Example item 5
7 Example item 6
8 Example item 7
9 Example item 8
10 Example item 9
11 Example item 10
12 Example item 11
13 Example item 12
14 Example item 13
15 Example item 14

Figure 3: CopyQ comes with eight default themes for visual customization of the software. These can also be edited later.



0 copyq2

1 b01_bilder

2 /home/dd/Pictures/copyq1.png

3 Tutorial – Shell Calendars

Command-line aficionados do not have to forgo calendars and appointment reminders. The shell offers many tools for user-friendly handling of date definitions in scripts.

4 Features

- Support for Linux, Windows and OS X 10.9+
- Store text, HTML, images or any other custom formats
- Quickly browse and filter items in clipboard history
- Sort, create, edit, remove, copy/paste, drag'n'drop items in tabs
- Add notes or tags to items
- System-wide shortcuts with customizable commands
- Paste items with shortcut or from tray or main window
- Fully customizable appearance
- Advanced command-line interface and scripting
- Ignore clipboard copied from some windows or containing some text
- Support for simple Vim-like editor and shortcuts
- Many more features

5 CopyQ monitors system clipboard

6 CopyQ is advanced clipboard manager with editing and scripting features.

7 https://hluk.github.io/CopyQ/

8

Figure 4: In the toolbar located on the right side of the main window(sporting the *forest* theme here), you will find the icons for editing the saved elements.

for quick access. Themes also allow the appearance of the main window to be largely freely determined (Figure 3).

But back to the entries themselves, which CopyQ refers to as elements: In the main window on the right side you will see a row of small icons that are used to manipulate the elements. If you mouse over them, a short label shows what they are used for. This ranges from manually creating an element to pinning elements to various predefined or self-created tags. Thus, elements can be activated for automated execution of commands, information about the file type of an element can be obtained, or elements can be removed if they are not pinned (Figure 4).

GUI, Keyboard, or CLI

In addition to the Edit menu, you can also use the keyboard to edit an element. With the element selected, pressing F2 opens the element for editing, and pressing F2 again saves and returns to the main window. Elements can be edited either in the main window or in the editor you set as default.

The software lets you tag or pin elements with predefined or DIY tags. You can also determine the slot in the hierarchy that an element will occupy. Phrases that you need all the time can be pinned to the top of the hierarchy.

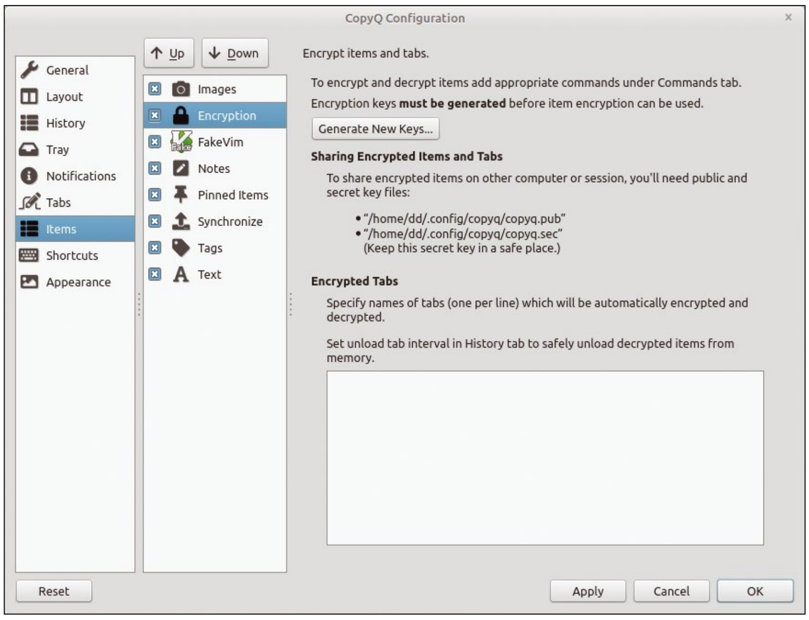
In the settings, you can alternatively enable controls like in Vim. The single and double arrow keys at the bottom are used to move elements up or down, or all the way up or down, respectively. If you are missing a search function right now, just start typing in the main window. The software highlights the found locations in color.

In addition to the everyday functions described so far, CopyQ has a lot of advanced features that would go beyond the scope of this article in their entirety. However, I will at least briefly mention the most important ones. For reference, please refer to the explanations in the documentation under *Advanced Topics*.

Professional Feature Set

Advanced options include protecting individual elements or entire tabs, which you can configure

Figure 5: CopyQ offers to encrypt individual items or entire tabs via GnuPG.



Encrypt items and tabs.

To encrypt and decrypt items add appropriate commands under Commands tab.
Encryption keys **must be generated** before item encryption can be used.

Sharing Encrypted Items and Tabs

To share encrypted items on other computer or session, you'll need public and secret key files:

- "/home/dd/.config/copyq/copyq.pub"
- "/home/dd/.config/copyq/copyq.sec"
  (Keep this secret key in a safe place.)

Encrypted Tabs

Specify names of tabs (one per line) which will be automatically encrypted and decrypted.

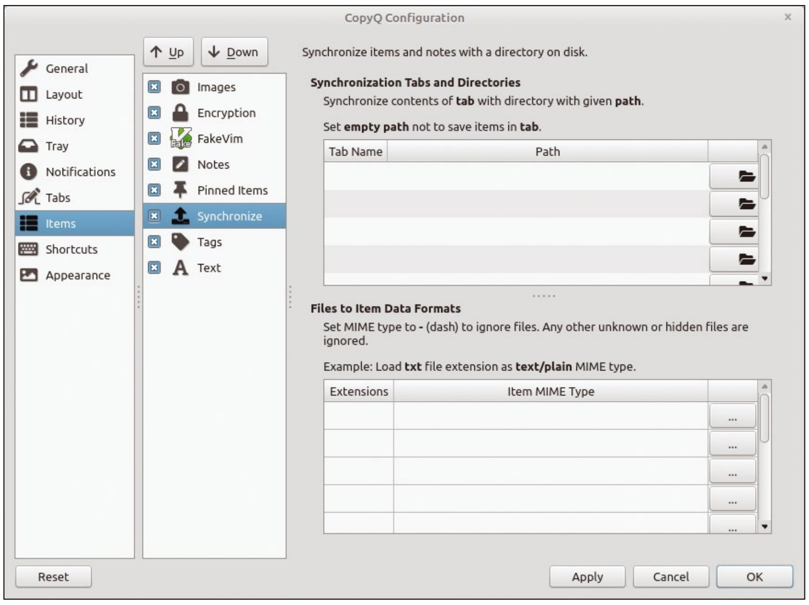
Set unload tab interval in History tab to safely unload decrypted items from memory.

in the preferences below *Items* | *Encryption*. In addition, items, notes, or tabs can be synchronized with a directory on the hard disk as a backup under *Items* | *Synchronize*, while the import function works in the opposite direction. If you want to protect content from tabs from unauthorized access, the software also lets you encrypt with GnuPG (Figure 5). Backups [3] offer a different form of security for the entire application (Figure 6).

You can enable predefined commands in CopyQ either in the main menu via *File* | *Commands/Global Shortcuts* or via F6. You can also add your own commands in the same way. These either act on new content in the clipboard, or you can execute them via the menu or keyboard shortcuts.

This can be used, for example, to integrate new commands into the menus, to automatically move URLs to their own tabs or to insert selected elements into external applications such as web browsers or image editors. The documentation provides further examples [4]. An API lets you run scripts to automate more complex operations [5]. Predefined commands [6] and scripts [7] for use with CopyQ can be found on GitHub.

All of CopyQ's functions can also be controlled by keyboard shortcuts. First of all, you will want to define one for the main window in the preferences below *Shortcuts*. A process manager that lists only CopyQ's processes can be called up in the main menu by selecting *File* | *Process Manager* or by using Ctrl+Shift+Z (Figure 7).



Synchronize items and notes with a directory on disk.

Synchronization Tabs and Directories

Synchronize contents of **tab** with directory with given **path**.

Set **empty path** not to save items in **tab**.

Tab Name	Path

Files to Item Data Formats

Set MIME type to **-** (dash) to ignore files. Any other unknown or hidden files are ignored.

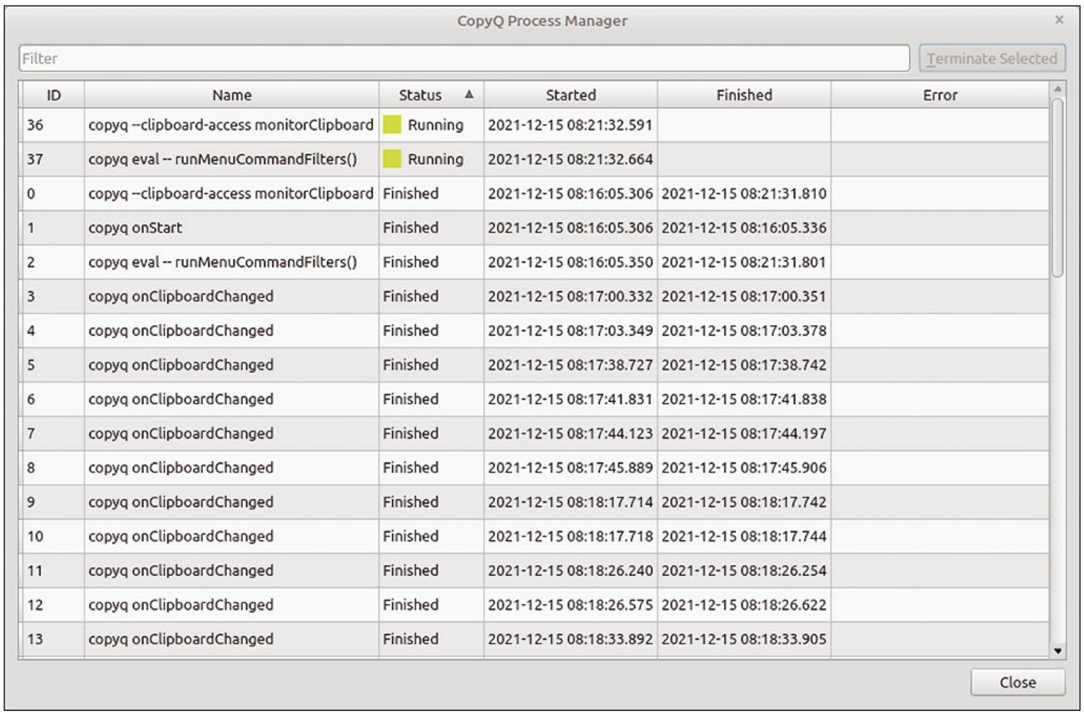
Example: Load **txt** file extension as **text/plain** MIME type.

Extensions	Item MIME Type

Figure 6: The software offers several options for backing up content. In addition to backing up the entire application via a script, individual tabs can also be mirrored to a directory on the hard disk.

Conclusions

CopyQ has outpaced the competition in recent years. It impresses with carefully considered functions that have supported me in my daily work for years, without any bugs annoying me in the process. Currently, CopyQ is probably the most powerful clipboard manager in the world. At the same time, the advanced functionality does not get in the way of the simple use of everyday functions anywhere. So, if you don't need the goodies, just ignore them. ■■■



ID	Name	Status	Started	Finished	Error
36	copyq --clipboard-access monitorClipboard	🟨 Running	2021-12-15 08:21:32.591		
37	copyq eval -- runMenuCommandFilters()	🟨 Running	2021-12-15 08:21:32.664		
0	copyq --clipboard-access monitorClipboard	Finished	2021-12-15 08:16:05.306	2021-12-15 08:21:31.810	
1	copyq onStart	Finished	2021-12-15 08:16:05.306	2021-12-15 08:16:05.336	
2	copyq eval -- runMenuCommandFilters()	Finished	2021-12-15 08:16:05.350	2021-12-15 08:21:31.801	
3	copyq onClipboardChanged	Finished	2021-12-15 08:17:00.332	2021-12-15 08:17:00.351	
4	copyq onClipboardChanged	Finished	2021-12-15 08:17:03.349	2021-12-15 08:17:03.378	
5	copyq onClipboardChanged	Finished	2021-12-15 08:17:38.727	2021-12-15 08:17:38.742	
6	copyq onClipboardChanged	Finished	2021-12-15 08:17:41.831	2021-12-15 08:17:41.838	
7	copyq onClipboardChanged	Finished	2021-12-15 08:17:44.123	2021-12-15 08:17:44.197	
8	copyq onClipboardChanged	Finished	2021-12-15 08:17:45.889	2021-12-15 08:17:45.906	
9	copyq onClipboardChanged	Finished	2021-12-15 08:18:17.714	2021-12-15 08:18:17.742	
10	copyq onClipboardChanged	Finished	2021-12-15 08:18:17.718	2021-12-15 08:18:17.744	
11	copyq onClipboardChanged	Finished	2021-12-15 08:18:26.240	2021-12-15 08:18:26.254	
12	copyq onClipboardChanged	Finished	2021-12-15 08:18:26.575	2021-12-15 08:18:26.622	
13	copyq onClipboardChanged	Finished	2021-12-15 08:18:33.892	2021-12-15 08:18:33.905	

Figure 7: The Process Manager, located under the File menu, shows all of CopyQ's processes since the application was last started, along with their statuses and potential errors.

Info

[1] CopyQ: *https://hluk.github.io/CopyQ/*

[2] CopyQ documentation: *https://copyq.readthedocs.io/en/latest/*

[3] Backup: *https://github.com/hluk/copyq-commands/blob/master/Scripts/backup-on-exit.ini*

[4] Command examples: *https://copyq.readthedocs.io/en/latest/command-examples.html#command-examples*

[5] Scripting API: *https://copyq.readthedocs.io/en/latest/scripting-api.html#scripting-api*

[6] Commands: *https://github.com/hluk/copyq-commands/blob/master/README.md*

[7] Scripts: *https://github.com/hluk/copyq-commands/tree/master/Scripts*

FOSSPicks

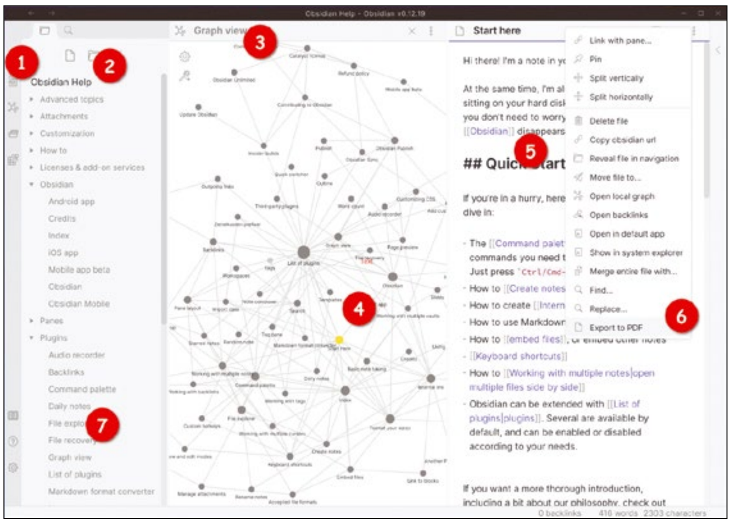
Sparkling gems and new releases from the world of Free and Open Source Software

This month, Graham is fighting off a new caffeine dependency after discovering his espresso machine can be hacked and automated by adding an Arduino! BY GRAHAM MORRISON

Knowledge editor

Obsidian

Linux is the worst choice of operating system for people who like to procrastinate because there's simply too much choice. There's also so much to learn when you do make a choice that it makes decisions even easier to postpone while you research another option. This is particularly true when it comes to note-writing applications because procrastinators are also often prolific note takers. There are so many ways to make your own notes that it's easier to spend time researching them than it is to make a decision and start using one. There are command-line tools, Markdown editors, editor add-ons such as Vimwiki and Emacs Org mode, and `echo` on the command line to a simple file. These all have their advantages, but none of them can easily link between notes and their wider context as you add links, make notes, and create different categories. This is a problem that Obsidian has been developed to solve, and it's already become hugely popular in the world of productivity hacking, thanks to it both supplanting other proprietary applications and being particularly good.



Hi there! I'm a note in yo

At the same time, I'm al
sitting on your hard disk
you don't need to worry
[[Obsidian]] disappears

Quick Start

If you're in a hurry, here
dive in:

- The [[Command palet
  commands you need t
  Just press `Ctrl/Cmd-
- How to [[Create notes
- How to create [[Intern
- How to use Markdown
- How to [[embed files]], or embed other notes
- [[Keyboard shortcuts]]
- How to [[Working with multiple notes|open multiple files side by side]]
- Obsidian can be extended with [[List of plugins|plugins]]. Several are available by default, and can be enabled or disabled according to your needs.

If you want a more thorough introduction, including a bit about our philosophy, check out

1. Markdown files: Your notes are saved as Markdown and the background intelligence comes from using an SQLite database. **2. Vaults:** Sets of notes are arranged in vaults, which are analogous to filesystem directories. You can also add plugins. **3. Graph view:** Maps each link from every note to the note they target, making it easy to see the relationship between your notes. **4. Node highlight:** Hover over a note node to see which notes it references and vice versa. **5. Editor:** The Markdown editor is simple, clean, and effective with an optional output preview. **6. Export:** Notes can be published directly or compiled into a PDF. **7. Help:** The help documentation has its own vault and is both substantial and well written.

Obsidian is a desktop-bound note-taking application built on Electron. It takes more system resources than nano, but you do get beautiful font rendering, spell checking, syntax highlighting, and excellent cross-platform compatibility, with AppImages for both ARM64 and x86, as well as a multitude of other install options. To start taking notes you first need to create a vault. A vault is like a folder or directory, and Obsidian includes a brilliant example *Help* vault containing its own documentation. With a vault started or loaded, Ctrl+N will create a new note and you can start typing. The notes themselves are written in simple Markdown so they can be read as raw text without difficulty, and they remain easily accessible from whichever editor you might want to use. The editing view can be split to show more than one note at a time, and a preview button lets you quickly see how they look rendered outside of the Markdown syntax. This can all be a little confusing in the beginning, but along with the excellent help documentation there's also a great hint system that will pop up to show you options when you start using commands and linking things. A command palette can also be opened to show all the shortcuts, and there's a Markdown bulk-importer that will convert some of the special formatting from rival applications. Markdown also means there's very little to learn about the syntax, except in the way Obsidian uses links.

The real genius of Obsidian is the way it's built around links, or connections. These are created by adding double square brackets to standard Markdown link syntax, creating back-links to pages within the vault. If a page doesn't exist, it can be created at the same time with a simple click. In this way you can quickly write not just your notes but also build a complex model of the relationships between one note and another without any further effort. Obsidian's best use of this feature is in the Graph view. This is a way to visualize your links, showing a node for each document and lines between nodes to illustrate the links. Mouse over a single note to highlight its connections and explore more relationships by filtering the nodes or animating their creation over time. It's a brilliant way of seeing which notes are interrelated and how they're organized, and it's a genuine reason to use Obsidian over any other note-taking tool you might otherwise be tempted by.

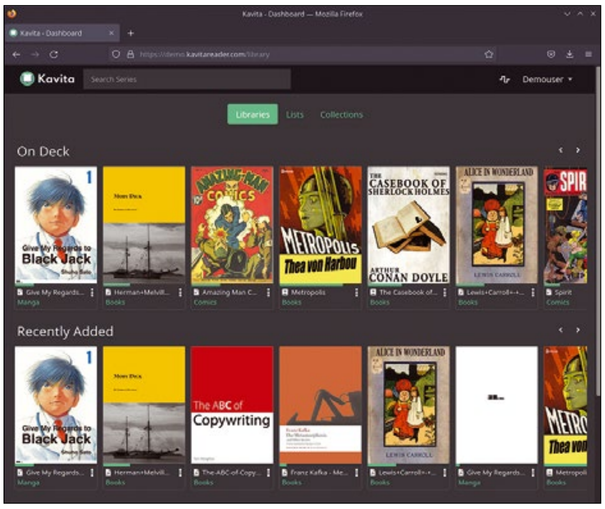
Project Website
https://obsidian.md

Collection manager

Kavita

There are many ways to track and manage a collection of things, from a simple file in your Documents folder or an open source database lovingly handcrafted from SQL to desktop applications or even spreadsheets. But there's nothing better than finding a tool that's been developed by someone with the same passion for a specific kind of collection. This is what Kavita is. It's a self-hosted, web-based collection manager built specifically for manga, comics, and books. It's not really limited to items like these, and you can create a collection out of anything, but text-based graphical items will be able to take advantage of integrated bookmarks, browser-based reading, and the search functionality, plus the image thumbnails used to view a collection will look more uniform.

The project has been built with the Angular framework using the Angular CLI. Angular itself is an interesting project, originally developed by Google to enable good-looking and complex web applications to be built with relative ease. Kavita includes a Linux installer to handle all these dependencies, and many people run the server within Docker. Whichever way you choose, it's easy to get set up. The web UI requires that you create a user account and you can then start your own library by importing your own titles. Kavita can consume manga as CBR, CBZ, ZIP/RAR, 7-Zip, and raw image files, and books as either EPub or PDF. You can view and read your titles from the web interface, create bookmarks, and continue a title from where you left off, even from a new session or different computer. You can rate items, add them to lists and categories, and obviously search across titles and collections for phrases you might be looking for. It works especially well on a tablet – where you can use the web interface just like an e-reader, only accessible from anywhere on your network – and leaves you wanting to start such a collection when you might not otherwise have considered it.



Kavita is designed to manage and collate your book and manga collection while also letting you read them from the same web interface.

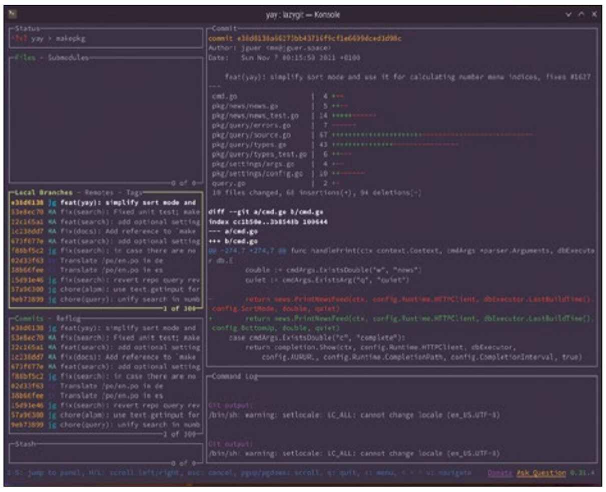
Project Website
https://github.com/Kareadita/Kavita

Git client

lazygit

Git starts off easy. You clone repositories, create branches, and push back changes to your shared code without troubling the documentation. It works like you might expect a version control system to work, and most projects can subsist like this, perhaps with a smattering of rebase and cherry picking. But with Git, it's only a matter of time before a fog of war descends over your project, turning your muscle memory commands into increasingly intricate knots that become almost impossible to untangle. It's then that you wish you'd used some kind of smart client to make better sense of it all. This is what lazygit attempts to be. Lazygit runs from, and takes over, your terminal, filling the area with five or six different panes. When first launched, it gamely asks you to watch a YouTube video explaining its features before presenting the latest release notes.

Despite all these panels, lazygit is easy to understand if you've used Git for some time. On the left, five panes show the current file and commit status, including modified files, a list of recent commit messages, and whether there are any stashed files (files from other branches stored for later retrieval). You can navigate between each pane with the tab key and through each separate entry with the cursors or Vim navigation keys. Selecting a file will update the right side of the view to show both the unstaged and staged changes for a file, with a diff viewing showing what's actually changed. This is one of the clearest interfaces we've seen for looking at the current state of a Git repository, and it gives you an immediate feel for what's being worked on and what has been recently completed. If you've not worked on a repository before, or not worked on a repository for some time, this overview helps you quickly get back up to speed.


```
yay : lazygit — Konsole

Status
(✓) yay → makepkg

Files - Submodules
                                   0 of 0

Local Branches - Remotes - Tags
e3bd0138 jg feat(yay): simplify sort mode and
53e8ec70 MA fix(search): Fixed unit test; make
22c165a1 MA feat(search): add optional setting
1c238dd7 MA fix(docs): Add reference to `make
673f077e MA feat(search): add optional setting
f88bf5c2 jg fix(search): in case there are no
02d33f63    Translate /po/en.po in de
38b66fee    Translate /po/en.po in es
15d91e46 jg fix(search): revert repo query rev
57a96300 jg chore(alpm): use text.getinput for
9eb73899 jg chore(query): unify search in numb
                                   1 of 300

Commits - Reflog
e3bd0138 jg feat(yay): simplify sort mode and
53e8ec70 MA fix(search): Fixed unit test; make
22c165a1 MA feat(search): add optional setting
1c238dd7 MA fix(docs): Add reference to `make
673f077e MA feat(search): add optional setting
f88bf5c2 jg fix(search): in case there are no
02d33f63    Translate /po/en.po in de
38b66fee    Translate /po/en.po in es
15d91e46 jg fix(search): revert repo query rev
57a96300 jg chore(alpm): use text.getinput for
9eb73899 jg chore(query): unify search in numb
                                   1 of 300

Stash
                                   0 of 0

Commit
commit e3bd0138a66173bb43716f9cf1e6639dced1d98c
Author: jguer <me@jguer.space>
Date:   Sun Nov 7 00:15:53 2021 +0100

    feat(yay): simplify sort mode and use it for calculating number menu indices, fixes #1627
---
 cmd.go                    |  4 +--
 pkg/news/news.go          |  5 ++--
 pkg/news/news_test.go     | 14 +++++------
 pkg/query/errors.go       |  7 -------
 pkg/query/source.go       | 67 ++++++++++++++++++++++---------------------------
 pkg/query/types.go        | 43 +++++++++++++++----------------
 pkg/query/types_test.go   |  6 +++---
 pkg/settings/args.go      |  4 +--
 pkg/settings/config.go    | 10 ++--------
 query.go                  |  2 +-
 10 files changed, 68 insertions(+), 94 deletions(-)

diff --git a/cmd.go b/cmd.go
index cc1b50e..3b8548b 100644
--- a/cmd.go
+++ b/cmd.go
@@ -274,7 +274,7 @@ func handlePrint(ctx context.Context, cmdArgs *parser.Arguments, dbExecuto
r db.E
        double := cmdArgs.ExistsDouble("w", "news")
        quiet := cmdArgs.ExistsArg("q", "quiet")

-       return news.PrintNewsFeed(ctx, config.Runtime.HTTPClient, dbExecutor.LastBuildTime(),
 config.SortMode, double, quiet)
+       return news.PrintNewsFeed(ctx, config.Runtime.HTTPClient, dbExecutor.LastBuildTime(),
 config.BottomUp, double, quiet)
    case cmdArgs.ExistsDouble("c", "complete"):
        return completion.Show(ctx, config.Runtime.HTTPClient, dbExecutor,
            config.AURURL, config.Runtime.CompletionPath, config.CompletionInterval, true)

Command Log

Git output:
/bin/sh: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)

Git output:
/bin/sh: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)

1-5: jump to panel, H/L: scroll left/right, esc: cancel, pgup/pgdown: scroll, q: quit, :: menu, ← → ↑ ↓: navigate   Donate Ask Question 0.31.4
```

One of the best things about the lazygit Git client, other than its name, is that it will perform an interactive rebase of your code repository.

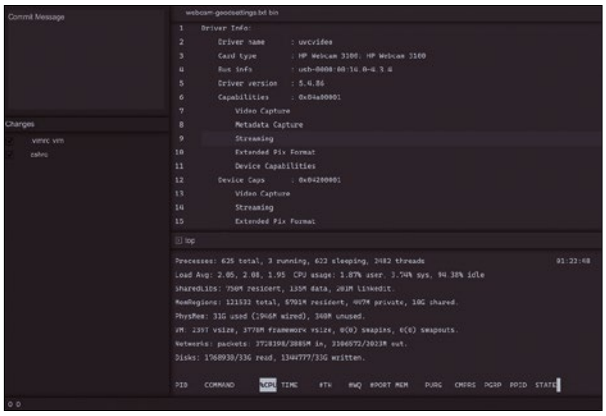
Project Website
https://github.com/jesseduffield/lazygit

Code editor

Lapce

We always joke that the reason why there are so many programming text editors is because creating such a text editor is an ideal first project for programmers, especially when they're switching to a new language. Because Rust seems to be a popular new language, there are lots of new editors written in Rust. Rust also seems to be one of those languages that's often mentioned specifically by programmers when they use it, much like Arch users never failing to mention their favorite Linux distribution. All of this is true of Lapce, a new programmer's editor that proudly wears its Rust editing credentials on its sleeve. But it does this in such a way that makes it worth the installation effort. In particular, it's fast, simple, and uses edit and command modes like Vim.

Lapce is still in its early developmental stage, but its creator has been using it as their daily driver for a year. The main window is sparse, but it's also fast and functional. There are four main panels: the main editing pane, an embedded terminal, a view for commit messages, and another for showing changed files. These all work best when you're within a Git repository, but of course there's nothing stopping you from changing things around. The editor is modal, which means there's an edit mode and a command mode. Like Vim, the command palette is opened with the : key, but with Lapce, you see all the commands you can summon, including those for opening files, creating new tabs, changing the theme (both light and dark), and viewing the configuration. There's also a WASI plugin system that's yet to be fully developed but should allow developers to port or create their own add-ons. It's early days, but Lapce already does enough to make it a good option for fast desktop editing, and it's likely to only get better in the future.



```
Driver Info:
    Driver name       : uvcvideo
    Card type         : HP Webcam 3100: HP Webcam 3100
    Bus info          : usb-0000:00:14.0-4.3.4
    Driver version    : 5.4.86
    Capabilities      : 0x84a00001
        Video Capture
        Metadata Capture
        Streaming
        Extended Pix Format
        Device Capabilities
    Device Caps       : 0x04200001
        Video Capture
        Streaming
        Extended Pix Format
```

```
Processes: 625 total, 3 running, 622 sleeping, 2482 threads                01:22:48
Load Avg: 2.05, 2.08, 1.95  CPU usage: 1.87% user, 3.74% sys, 94.38% idle
SharedLibs: 750M resident, 135M data, 201M linkedit.
MemRegions: 121533 total, 5791M resident, 497M private, 10G shared.
PhysMem: 31G used (1946M wired), 340M unused.
VM: 2397 vsize, 3778M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 2728198/3885M in, 3106572/2023M out.
Disks: 1768930/33G read, 1344777/33G written.

PID   COMMAND     %CPU TIME     #TH   #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE
```

In the background, Lapce talks to its own proxy that manages file sync and saves – which is quite unique.

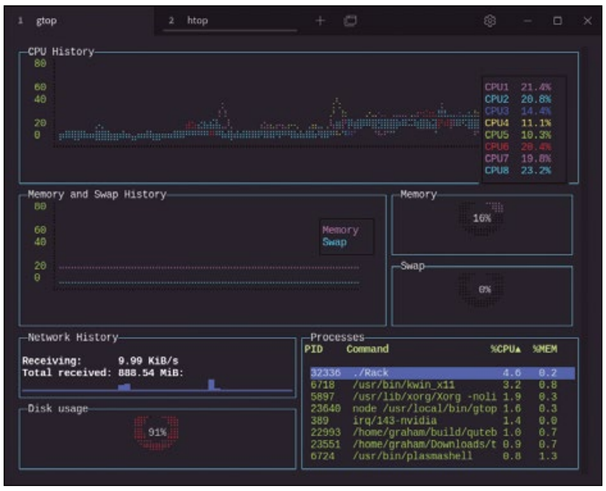
Project Website
https://github.com/lapce/lapce

Terminal emulator

Tabby

We're used to seeing new terminal emulators appear but not for terminal emulators to appear fully formed and ready to use. Yet this is the case with the brilliantly named Tabby, both for the key you frequently use on the command line and because, we presume, it's a name not dissimilar to that other flashy new terminal emulator, Kitty. But there's a good reason why Tabby has appeared almost fully formed and perfectly usable from day one, and that is because it was already an established project under the name of Terminus. Tabby hopes, as did Terminus, to be a terminal emulator for the modern age. In being so, Tabby offers lots of performance and convenience features that your default terminal from the 1970s might otherwise be lacking. These differences don't make a huge difference on Linux, but they might if you want to use the same terminal across various platforms. Tabby can be easily installed on WSL with PowerShell, Cygwin, and Git Bash, as well as binary clients for Windows, macOS, and of course Linux.

The next reason to use Tabby is convenience. From the first launch, there's very little setup required to create a functional environment. Tabby includes its own SSH and Telnet connection managers, with SFTP and port forwarding, as well as a serial terminal for more conveniently hacking on devices. The main window can split and host multiple sessions with tabs on either side, which is great for both left- and right-handed people, and the tabs themselves can be nested and configured to show progress and notifications. There's also a "quake" mode for sliding the terminal onto a screen edge, much like Yakuake, but without the configuration and the KDE. The same is also true of setting global hotkeys. Plus you can add functionality with plugins and hide secrets in Tabby's own SSH vault. Tabby might not be as good as a fully configured Konsole, but it's better out of the box, especially on non-Linux systems.



```
CPU1  21.4%
CPU2  20.8%
CPU3  14.4%
CPU4  11.1%
CPU5  10.3%
CPU6  20.4%
CPU7  19.8%
CPU8  23.2%

Receiving:       9.99 KiB/s
Total received:  888.54 MiB:

PID     Command                    %CPU▲  %MEM
32336   ./Rack                     4.6    0.2
6718    /usr/bin/kwin_x11          3.2    0.8
5897    /usr/lib/xorg/Xorg -noli   1.9    0.3
23640   node /usr/local/bin/gtop   1.6    0.3
389     irq/143-nvidia             1.4    0.0
22993   /home/graham/build/quteb   1.0    0.7
23551   /home/graham/Downloads/t   0.9    0.7
6724    /usr/bin/plasmashell       0.8    1.3
```

In addition to providing dozens of small functions, Tabby is easily themed to fit whichever environment and operating system you're using.

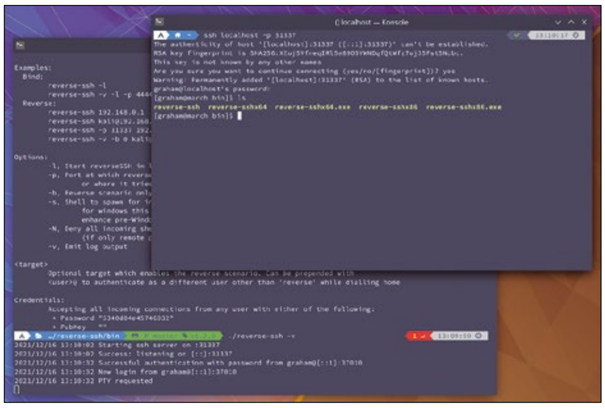
Project Website
https://github.com/Eugeny/tabby

Security threat

ReverseSSH

This is a bit of a cautious find because its most obvious use is to help people gain unauthorized access to a network, which is an action we absolutely don't condone. But neither do we condone security through obscurity because, in our experience, it's always helpful to know what kinds of tools are out there and what they do, especially when they have actual utility. This is the case with ReverseSSH, a project with a particularly descriptive name. ReverseSSH is a modified version of the same SSH server binary that many of us rely on every day to make connections between our various servers, Raspberry Pis, and home automation systems. ReverseSSH has been modified to allow anyone with physical access to a system, and presumably a way to get the binary onto that system, to run the modified server in such a way that an external SSH connection can be made and connected automatically, albeit with a single given password.

This is obviously bad for network security, but if you're worried about such things on your network, there are far more nefarious and less obvious attack vectors, such as manually setting up reverse SSH with netcat. The best way of defending against such things is to limit access to your network. With that out of the way, ReverseSSH can also be a helpful tool because it's a single, easily executable command (`reverse-ssh`) that enables remote access to a system, with no further configuration. Run the executable on one machine and use SSH on another to access the system. You can then use the shell, transfer files with SFTP, set up port forwarding, and do all the other fun things that SSH allows, all without worrying about security. This can be helpful on a home network or when you absolutely need to accept all incoming connections without the risk of authentication – when used with extreme caution.



```
ssh localhost -p 31337
The authenticity of host '[localhost]:31337 ([::1]:31337)' can't be established.
RSA key fingerprint is SHA256:XEuj5YfreqIRl5e89O5YWNDqfQtWfz7ej35fsts5NLbc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:31337' (RSA) to the list of known hosts.
graham@localhost's password:
[graham@march bin]$ ls
reverse-ssh  reverse-sshx64  reverse-sshx64.exe  reverse-sshx86  reverse-sshx86.exe
[graham@march bin]$ 
```

```
Examples:
  Bind:
        reverse-ssh -l
        reverse-ssh -v -l -p 4444
  Reverse:
        reverse-ssh 192.168.0.1
        reverse-ssh kali@192.168.
        reverse-ssh -p 31337 192.
        reverse-ssh -v -b 0 kali

Options:
        -l, Start reverseSSH in l
        -p, Port at which revers
                or where it trie
        -h, Reverse scenario only
        -s, Shell to spawn for in
                for windows this
                enhance pre-Wind
        -N, Deny all incoming she
                (if only remote
        -v, Emit log output

<target>
        Optional target which enables the reverse scenario. Can be prepended with
        <user>@ to authenticate as a different user other than 'reverse' while dialling home

Credentials:
        Accepting all incoming connections from any user with either of the following:
         * Password "5340d84e45746032"
         * PubKey   ""
./reverse-ssh/bin  ./reverse-ssh -v
2021/12/16 13:10:02 Starting ssh server on :31337
2021/12/16 13:10:02 Success: listening on [::]:31337
2021/12/16 13:10:32 Successful authentication with password from graham@[::1]:37010
2021/12/16 13:10:32 New login from graham@[::1]:37010
2021/12/16 13:10:32 PTY requested
```

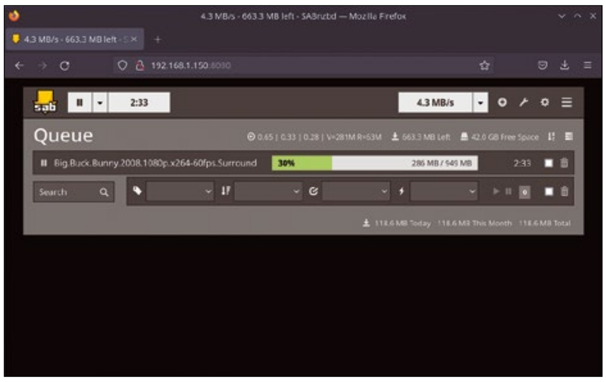
Use the help output to discover the password used to connect to all sessions, which is unique when you build it yourself.

Project Website
https://github.com/Fahrj/reverse-ssh

Usenet downloader

SABnzbd

Back in the mists of time, before Facebook, Twitter, and even Google, Internet users got their news from "newsgroups." These were secretive enclaves accessed from the terminal where anyone could post, read, and reply to loosely related articles and comments. The great thing about newsgroups was that they were completely decentralized with a web of interconnected servers bouncing messages between themselves. Early ISPs would even have their own news servers, creating an early form of CDN, so that their users could have access to a high performing local news cache. You might reasonably question why this was needed for simple text, but by then text had been supplanted by binaries, albeit encoded as text with `uuencode`, and Usenet became the go-to location for any kind of file.
Remarkably, Usenet still exists as a kind of hinterland of the old Internet. There are still lots of conversation groups, but there's equally still every kind of binary. It can often be the only place to find old music or Amiga files that were once hosted freely. There's now very little software that can access the old network, but SABnzbd is one that can, built specifically for retrieving those illicit binary files via an aggregation of the posts you want within an NZB file. It's a project that's been around for many years, but it's still being developed and has only recently made the transition from Python 2.x to Python 3.x with its own 3.0 and 3.5 releases. When it's running, you can add NZB files using a variety of methods, from watching a directory to setting up an RSS feed, and downloads are automatically grabbed, decompressed, and repaired if necessary. There are many configuration options, from the width of the web UI to API key access, and it runs perfectly on anything from a Raspberry Pi or NAS to a cloud instance somewhere.



The first rule of Usenet is that no one should mention Usenet. Oops!

Project Website
https://sabnzbd.org

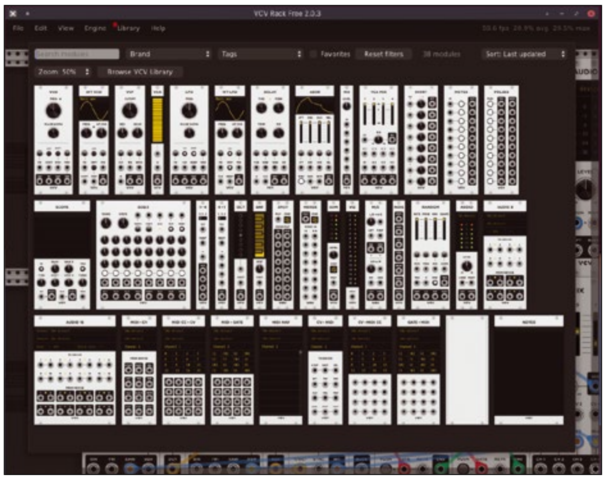
Modular synthesizer

VCV Rack 2

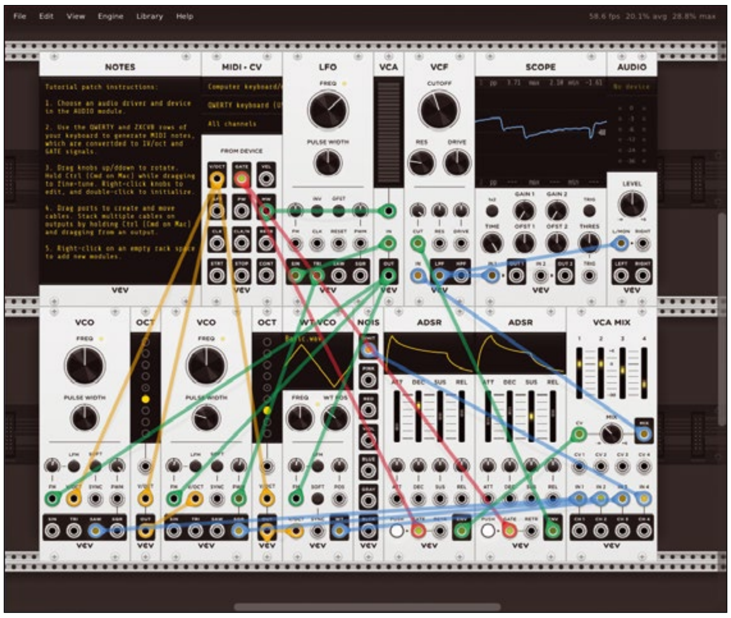
There are few open source projects that not only broaden the possibilities for their users but also kickstart an entire software ecosystem. Blender is one of those projects, as is Krita, but so too is the amazing VCV Rack (now often abbreviated to Rack). VCV Rack is a virtual modular synthesizer where different modules can be connected to one another to create an infinite variety of sounds, effects, drums, noises, note sequences, controller responses, and physical devices. What was most remarkable about this project was that it spawned a huge library of first- and third-party modules that could be added to your virtual racks, many of them modeled on real hardware that might cost hundreds or thousands of dollars but instead given away for free in their virtual form. There was a real synergy between certain hardware vendors, such as Mutable Instruments, who purposefully built open source hardware and firmware, and the developers who created modules based on their hardware to publish either exact copies of the originals or modules that were inspired by their functionality.

This plug-in system enabled anyone to create their own modules, and lots of people did. With VCV Rack, any of these modules can be dragged into your virtual rack and wired up with virtual cables, all hanging together in virtual gravity with the sound piped to your headphones. Their prevalence was enough to spawn a separate third-party ecosystem, with some module creators charging for their own creations and others requiring complex licenses to cover their mixture of code. All of this became tied to an associated VCV online account and store framework which provided access to these modules and managed those you could purchase. All this success also brought a little drama. When it was clear the original project was working towards accounts to enable paid-for modules and eventually a Pro version, which has finally appeared with the release of VCV Rack 2, forks were made for Apple's macOS and iOS and plug-in versions that worked with digital audio workstations, all created and maintained by independent developers, though most have now fallen by the wayside. These developments created some friction both in the community and with some open source module developers, but the bottom line was that thanks to its popularity, any new version would always be open source (GPLv3) and contain a staggering number of updates. And this is exactly what's happened with the release of version 2, albeit alongside the commercial Pro version that can now also work as a plug-in.

A new graphics engine in version 2 still needs the help of a GPU, but there's now a "dark room mode" to help all those middle-aged synth tinkerers work through the night. You can also more easily save, edit, and share your module selection, albeit through the account system, and it's now simpler to find the module you're looking for with the new module browser. All of this is still open source and available for free with the standalone version. The high quality of this update is likely only possible because the commercial side of the project fuels its open source development, and that has to be a good thing. There are now more than 2,000 modules to choose from, with more than 170 from official module manufacturers, and 30+ integrated into the default. Version 2 has just been released to hopefully build on this success and take it even further. If other people don't like these improvements, or the modules, or the direction the project is going, they can obviously still do whatever the GPLv3 allows them to do with the code, but there's been no doubt that this success has hugely helped fuel the ambitious VCV Rack 2.



Unlike real Eurorack modulars, VCV Rack can also be polyphonic – playing more than one note at a time from the same unit.



Tutorial patch instructions:

1. Choose an audio driver and device in the AUDIO module.

2. Use the QWERTY and ZXCVB rows of your keyboard to generate MIDI notes, which are converted to 1V/oct and GATE signals.

3. Drag knobs up/down to rotate. Hold Ctrl (Cmd on Mac) while dragging to fine-tune. Right-click knobs to edit, and double-click to initialize.

4. Drag ports to create and move cables. Stack multiple cables on outputs by holding Ctrl (Cmd on Mac) and dragging from an output.

5. Right-click on an empty rack space to add new modules.

VCV Rack 2 allows you to create a virtual modular synthesizer of any size without incurring any cost.

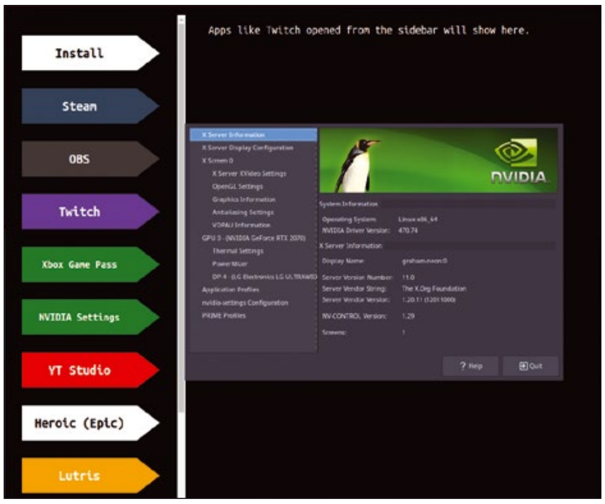
Project Website
https://vcvrack.com/

Setup script

Gamebuntu

This is an exciting time for Linux gamers because so much is happening to make gaming on Linux better. This is of course mostly thanks to Valve, which has been lavishing attention on Linux, Wine, and its Proton compatibility layer in preparation for the release of its Steam Deck handheld gaming PC. But there are plenty of other developments too, and Gamebuntu is one of them. It's the brainchild of Rudra Saraswat, a developer based in India who's also responsible for the Ubuntu Unity Remix distribution and, at the time of writing, is still only 12 years old! Gamebuntu is a comprehensive Bash script with a graphical front end that aggregates the installation of everything and anything related to gaming. It's designed for people new to Linux who might install something like Ubuntu but not know, or want to know, what else should be added to create the perfect gaming system.

The graphical front end starts by showing an *Install* button in the sidebar in the top left of the window. Clicking this will open a terminal that automatically installs a huge pile of gaming-related software. This can take some time because the list of what's installed is huge and continually growing. It includes things such as the XanMod Kernel for better gaming performance, a Mesa PPA to access the latest graphics drivers, OBS with PulseAudio effects, and NoiseTorch for game streaming with microphone noise reduction. For gaming, Gamebuntu adds Steam, Lutris, and Heroic Games Launcher to access your games, and Wine for playing Windows titles. When the script completes, the front end becomes a launcher to access all the user-facing parts of everything that has been installed. This may require some extra setup, such as logging in for the Epic Games Store or YouTube Studio access, but it quickly becomes an excellent way to access all your gaming-related content and is a strong reminder of just how much high-quality gaming-related software there is available for Linux, with hopefully much more to come.



Apps like Twitch opened from the sidebar will show here.

X Server Information
X Server Display Configuration
X Screen 0
X Server XVideo Settings
OpenGL Settings
Graphics Information
Antialiasing Settings
VDPAU Information
GPU 0 - (NVIDIA GeForce RTX 2070)
Thermal Settings
PowerMizer
DP-4 - (LG Electronics LG ULTRAWID
Application Profiles
nvidia-settings Configuration
PRIME Profiles

System Information
Operating System: Linux-x86_64
NVIDIA Driver Version: 470.74
X Server Information
Display Name: graham-neon:0
Server Version Number: 11.0
Server Vendor String: The X.Org Foundation
Server Vendor Version: 1.20.11 (12011000)
NV-CONTROL Version: 1.29
Screens: 1

The list of software installed automatically by Gamebuntu is changing all the time, with recent updates adding Twitch, Kodi, and VLC.

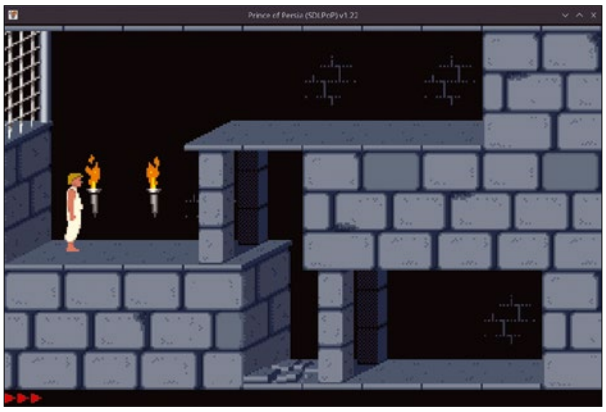
Project Website
https://gitlab.com/rswat09/gamebuntu

Prince of Persia clone

SDLPoP

Prince of Persia is one of the best 2D platform and adventure games of all time. Originally released in 1989 at the end of the 8-bit era and developed by Jordan Mechner on an Apple II, the game was remarkable for several reasons. The first was its realistic character animation. The sideways animations for the main character and his various enemies were all drawn by hand from actual video footage of Jordan Mechner running, jumping, and sword fighting. This gave the game an uncanny realism that was only matched when motion capture became prevalent many years later. Another reason the game was so compelling was its game play. You start in a palace dungeon with only 60 minutes to escape and save the heroine. Through countless hours of play testing, the game difficulty and playability was fine-tuned perfectly, making the quest not too difficult but not too easy either. The gaps between platforms were perfectly placed, as were the enemies and challenges in finding your way out. There was just enough variety and challenge to keep you hooked, as many of us were.

While the original game has never been released as open source, and the franchise continues on modern hardware, the hand-assembled original has now been decompiled and the resultant code rebuilt using SDL, creating SDLPoP. It's a perfect recreation of the original game, including the graphics, sound, music, and immortal gameplay. Even now, with the game more than 30 years old and modern games bundling ray tracing, AI, and hundreds of simultaneous online players, Prince of Persia is a lot of fun to play. This version also includes cheats (skip level, room navigation, kill and resurrection keys, and slow falling), mod support, and the ability to record video replay videos and save games, which also make it more accessible to a younger generation – or perhaps to those of us without the same wits and agility of our youth.



Prince of Persia has some great sword fighting sequences, modeled on Errol Flynn and Basil Rathbone in the 1938 film *The Adventures of Robin Hood*.

Project Website
https://github.com/NagyD/SDLPoP

Keeping tabs on your photo library with PhotoPrism

Machine Learning Smarts for Shutterbugs

PhotoPrism offers a combination of a polished, user-friendly interface and an artificial intelligence engine that makes organizing, searching, and sharing photos a breeze.

BY DMITRI POPOV

Organizing a photo library containing thousands of RAW files, photos, and videos can be a time-consuming and mind-numbing task. The mere thought of manually tagging, grouping, and organizing photos and videos can give even the most patient photographers shivers. Good thing then that we live in the age of artificial intelligence (AI) that can take the burden of boring tasks off our shoulders. For anyone who is not keen on spending countless evenings fighting a losing battle of keeping their photo libraries neatly organized, PhotoPrism [1] might be manna from heaven. This application pairs a polished yet capable user interface with the powerful TensorFlow machine learning engine, providing you with a platform that can take care of keeping tabs on your photos and videos as if by magic. That description is not an exaggeration: The inner workings of TensorFlow are hidden away from the user's sight so well that it's almost too easy to forget that PhotoPrism is powered by a complex piece of software engineering and not fairy dust. Of course, TensorFlow is not perfect, and sometimes it produces some hilarious results. But they are pretty much the only telltales of what actually powers PhotoPrism.

TensorFlow and PhotoPrism

PhotoPrism relies on TensorFlow to perform three important tasks. The first task is image classification. To simplify, TensorFlow analyzes images and assigns relevant labels to them. For example, all architectural photos get the *Building* label, and wildlife photos may get various labels, depending on the main subject (for example, *Bird*, *Butterfly*, etc.). The second task is face recognition, which allows PhotoPrism to find similar faces and group them into clusters. The third task is detecting images that are usually referred to as "not safe for work," which in most cases means photos containing nudity.

Deploying PhotoPrism

Because PhotoPrism consists of several complex moving parts, deploying it would be a rather daunting proposition if it weren't for containers. PhotoPrism is distributed as a Docker container image that reduces the task of getting the application up and running to a few simple steps.

The first order of business is to install Docker and Docker Compose on the machine that you plan to use for running PhotoPrism. If the machine happens to run an Ubuntu-based Linux distribution, installing both packages is a matter of issuing the command:

```
sudo apt install docker docker-compose
```

With the required software installed, use

```
wget https://dl.photoprism.org/docker/⮐
 docker-compose.yml
```

to fetch the configuration file.

While you can start PhotoPrism right away, it's a good idea to adjust the application's default settings first. To do this, open the `docker-compose.yml` file for editing. Most of the settings in the file are best left at their default values, but there are a few options that require tweaking. By default, PhotoPrism is configured to run in the private (password-protected) mode. If you want to keep it that way, you should replace the default password with a stronger one by replacing the `PHOTOPRISM_ADMIN_PASSWORD` variable. But if you plan to run the application on a machine that is not exposed to the Internet, and you feel confident that your PhotoPrism instance doesn't need any protection, enable the public mode by setting `PHOTOPRISM_PUBLIC` to `true`.

And, of course, you might want to replace the generic name and description of your PhotoPrism instance by editing the `PHOTOPRISM_SITE_TITLE`, `PHOTOPRISM_SITE_CAPTION`, `PHOTOPRISM_SITE_DESCRIPTION`, and `PHOTOPRISM_SITE_AUTHOR` options.

By default, PhotoPrism uses the `~Photos` directory as the location of your photo library. But if you store images and videos in a different location, you need to point PhotoPrism to it. To do this, locate the following line in the `docker-compose.yml` file:

```
- "~/Pictures:/photoprism/originals"
```

Replace `~/Pictures` with the absolute path to the library as follows:

```
- "/path/to/library:/photoprism/originals"
```

But what if you don't keep all your files in just one directory? It's not unusual to keep videos in a separate directory or even on a different storage device, and you might prefer to keep casual snaps in a dedicated library. PhotoPrism makes it possible to include other directories as subdirectories in the `originals` directory. To do this, specify additional entries under the `volumes` section as follows:

```
volumes:
  - "/path/to/videos:/photoprism/originals/videos"
  - "/path/to/snaps:/photoprism/originals/snaps"
```

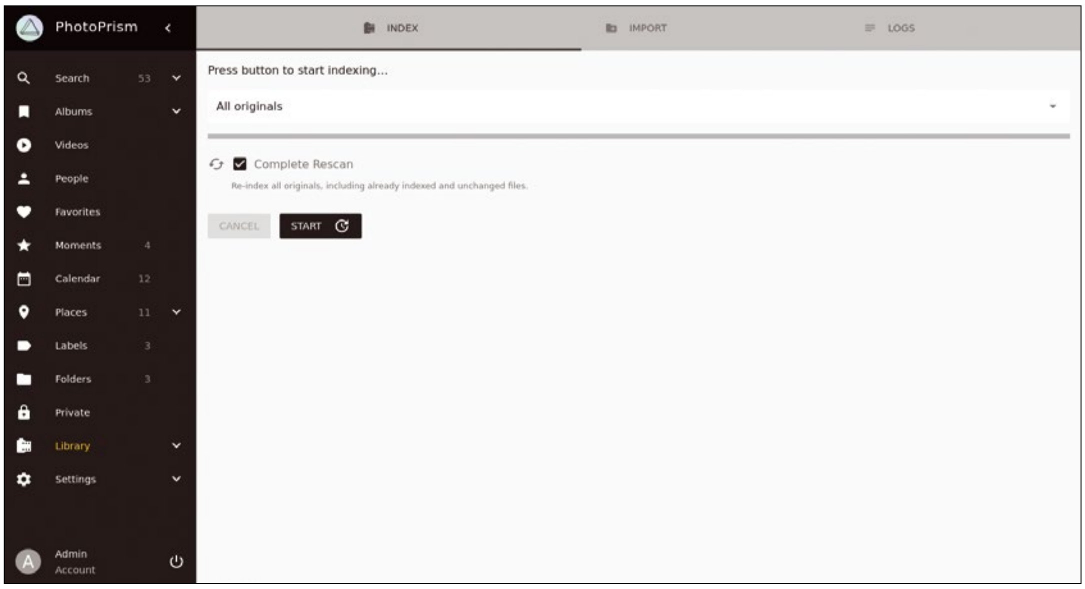
Each option in `docker-compose.yml` contains a description that can help you decide whether you want to modify it or leave it as is. Once you're done editing the configuration file, save the changes, and you're almost ready to start PhotoPrism -- almost because there is one more thing you might want to do. By default, PhotoPrism supports a wide range of file formats, and the application indexes all compatible files by default. But that might not necessarily be what you want. For example, if each RAW file in the library has a matching JPEG, there is probably little point in indexing the RAW original. To exclude specific files from being indexed, create a `.ppignore` file in the root of your photo library, and specify the desired pattern in the file. You can use wildcards when defining patterns, so adding the `*.NEF` pattern will exclude all NEF files from being indexed. Patterns specified in the `.ppignore` file apply both to the directory the file is in and all subdirectories. When you place the file in the root directory, the patterns specified in the file will be applied to the entire library. You can, however, place the file into any subdirectory, leaving all directories above it unaffected.

Now you can launch PhotoPrism by running

```
docker-compose up -d
```

as root in the directory that contains the `docker-compose.yml` file. Assuming you haven't changed the default port in the configuration file, point the browser to the *127.0.0.1:2342* address (replace *127.0.0.1* with the actual IP address of the machine running PhotoPrism), and log in using the *admin* username and the password you've specified in the configuration file.

Now that PhotoPrism is up and running, you need to run an indexing operation (Figure 1) that processes all RAW and JPEG files as well as videos (excluding, of course, the files specified in the `.ppignore` file). To do this, switch to the *Library* section and press *Start* to initiate the indexing operation. This task may take a while to complete, depending on the number of files in your library and the hardware PhotoPrism is running on. That's one of the reasons why you'd want to exclude certain files from being indexed – this may significantly reduce the indexing time. Keep in mind that you have to perform indexing every



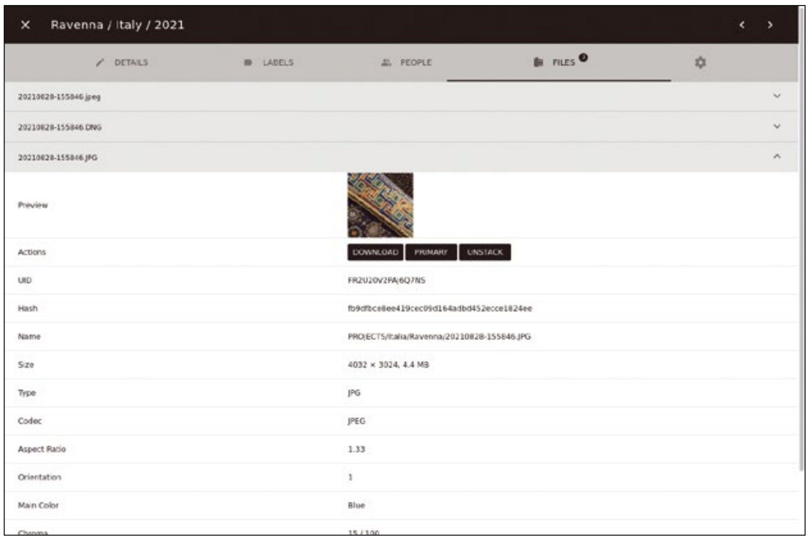
Press button to start indexing...

All originals

Complete Rescan

Re-index all originals, including already indexed and unchanged files.

Figure 1: To populate PhotoPrism with photos and videos, you need to index your library first.



Ravenna / Italy / 2021

DETAILS LABELS PEOPLE FILES SETTINGS

20210828-155846.jpeg

20210828-155846.DNG

20210828-155846.JPG

Preview	
Actions	DOWNLOAD PRIMARY UNSTACK
UID	FR2U20V2FA|6Q7NS
Hash	fb9dfbce8ee419cec09d164adbd452ecce1824ee
Name	PROJECTS/Italia/Ravenna/20210828-155846.JPG
Size	4032 × 3024, 4.4 MB
Type	JPG
Codec	JPEG
Aspect Ratio	1.33
Orientation	1
Main Color	Blue
Chroma	15 / 100

Figure 2: PhotoPrism groups photos into stacks. You can manually choose which image should be the primary one.

time you add new files to the library. Don't forget to disable the *Complete Rescan* option when indexing newly added files; otherwise, PhotoPrism will perform full re-indexing of the entire library.

During indexing, PhotoPrism not only classifies files but also groups related images into so-called stacks. For example, if you have a RAW file `19700101-112501.RAW`, its JPEG version `19700101-112501.JPG`, and processed variant `19700101-112501.jpeg`, PhotoPrism conveniently stacks them together, with the `19700101-112501.JPG` file at the top of the stack (i.e., this is the file PhotoPrism shows you). Stacks in PhotoPrism are identifiable by the icon in their upper-left corner. If the stack contains a RAW file, you can view it by clicking on the *Camera* icon, and if the stack consists of files in JPEG and other supported formats, you can view all of them by clicking on the *Stack* icon (Figure 2).

That's all fine and good, but what if you want to have another file in the stack at the top? What if in the example above, you'd rather see the processed `19700101-112501.jpeg` file than the unprocessed `19700101-112501.JPG` one? To do this, click on the image to open it in the view mode, then click on the *Edit* icon in the toolbar at the top. Switch to the *Files* section (its title shows the number of files in the stack), click on the desired file item, and press the *Primary* button in the *Actions* section. Exit the editing mode, refresh the page, and you should see the selected file at the top of the stack.

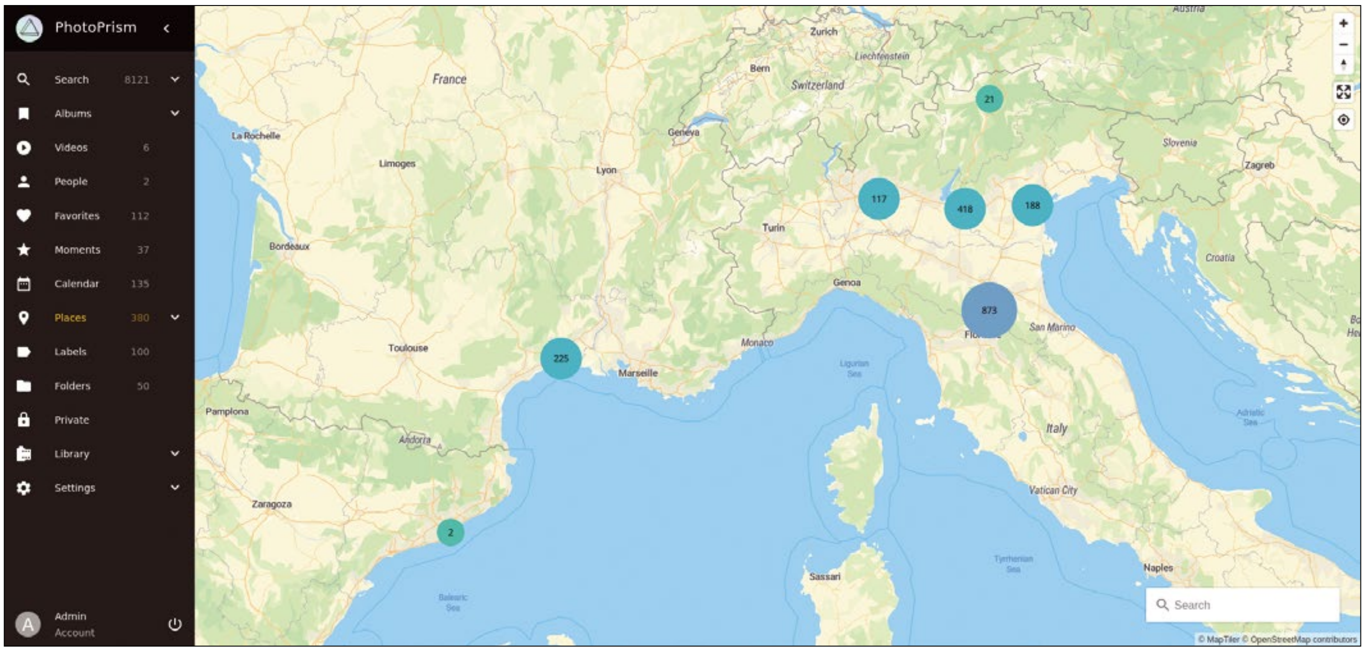
Browse and Search

In many respects, PhotoPrism behaves like a regular photo management application. The *Folders* module provides access to the library directories, the *Calendar* module displays all indexed files grouped by year and month, and the *Places* module presents you with a map where all photos and videos are displayed as clusters (Figure 3). While you can use the mouse to drag the map around to locate the desired spot and then zoom in and out, you can quickly jump to the desired location using the *Search* field. Type the name of the desired place (e.g., *Ravenna*) and hit *Enter* to jump to the specified location.

PhotoPrism groups all similar faces into clusters, and you can view and manage them in the *People* module. Because PhotoPrism does all the heavy lifting of detecting and grouping faces during the indexing operation, you only need to add names to the created clusters.

The *Moments* module provides a slightly different take on presenting photos from the past. Instead of the more conventional approach of showing photos taken on the current date in previous years, PhotoPrism uses an algorithm that creates albums based on a significant number (Figure 4) of photos taken in a certain year and in a certain place. For example, if in 2017 you took a significant number of photos on

Figure 3: *Places* shows all the photos in the library as clusters on a map.



your trip to Japan, PhotoPrism will create an album for them in the *Moments* module. In addition to that, the algorithm also finds labels with a significant number of photos and creates albums in the *Moments* module based on the results. The "significant number" value is determined dynamically, based on the overall number of items in the library.

Navigating the library by traversing the available categories provides a convenient way to quickly locate photos that you already know where to find. But that, of course, is not always the case. This is where PhotoPrism's search functionality comes into play, and the *Search* section gives you access to all the search features the application has to offer. Unsurprisingly, the most straightforward way to run a search is to type the desired search term in the *Search* field and hit Enter. This way, you can search for people, objects in the photos, filenames, locations, and even the dominating color (Figure 5). PhotoPrism also makes it possible to filter photos using several predefined criteria, such as country, year,

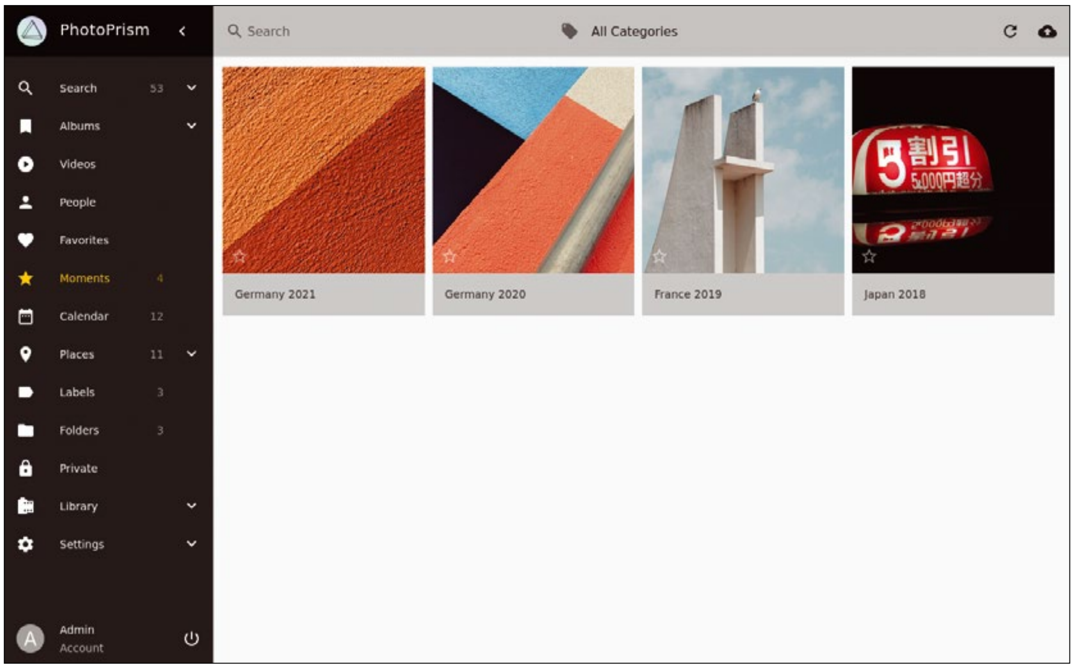


Figure 4: The *Moments* module determines the most important moments in your life by the number of photos you took.

camera, and lens. To access the filters, click on the *Expand Search* triangle on the right side of the search bar and select the desired filter or filters.

For more advanced and fine-grained searches, you can create advanced search queries using the available filters. For example, to find all photos taken after a certain date, specify the *after* filter in the search field as follows: *after:1970-01-31*. You can specify multiple filters. To find, for example, photos in a certain country before a specific date, you can combine the *before* and *country* filters: *country:de before:1970-01-31*. Search queries can include the OR operator. The query *color:red|green* will find photos with either red or green as the dominant color. The *name* filter makes it possible to search for filenames, and you can use wildcards with it. PhotoPrism gives you a wide range of filters, and the dedicated documentation page [2] contains a list of all supported filters along with their descriptions and examples.

Housekeeping

Because PhotoPrism runs in a container, there are a handful of commands you might find useful for managing the application. You already know

```
docker-compose up -d
```

starts PhotoPrism. To stop the application, use

```
docker-compose stop
```

If you need to reset PhotoPrism and start from scratch, use:

```
docker-compose exec photoprism photoprism ⮐
  reset
```

Finally, upgrading PhotoPrism is a matter of running the following commands:

```
docker-compose pull photoprism
docker-compose stop photoprism
docker-compose up -d photoprism
```

Wrap-Up

Relegating classification and face recognition tasks to the mighty TensorFlow engine is what sets PhotoPrism apart from pretty much everything else. Does it produce perfect results? Definitely not. But the sheer convenience of being able to plug your entire library into PhotoPrism and let the application do the rest outweighs occasional misses. It would be a mistake to judge PhotoPrism by its machine learning smarts only, though. The application's polished and user-friendly interface coupled with excellent search capabilities makes it a perfect tool for managing large libraries and sharing your photos with the rest of the world. ■■■

Info

[1] PhotoPrism: *https://photoprism.app/*

[2] PhotoPrism search: *https://docs.photoprism.org/user-guide/organize/search/*

The Author

Dmitri Popov has been writing exclusively about Linux and open source software for many years. His articles have appeared in Danish, British, US, German, Spanish, and Russian magazines and websites. You can find more on his website at *tokyoma.de*.

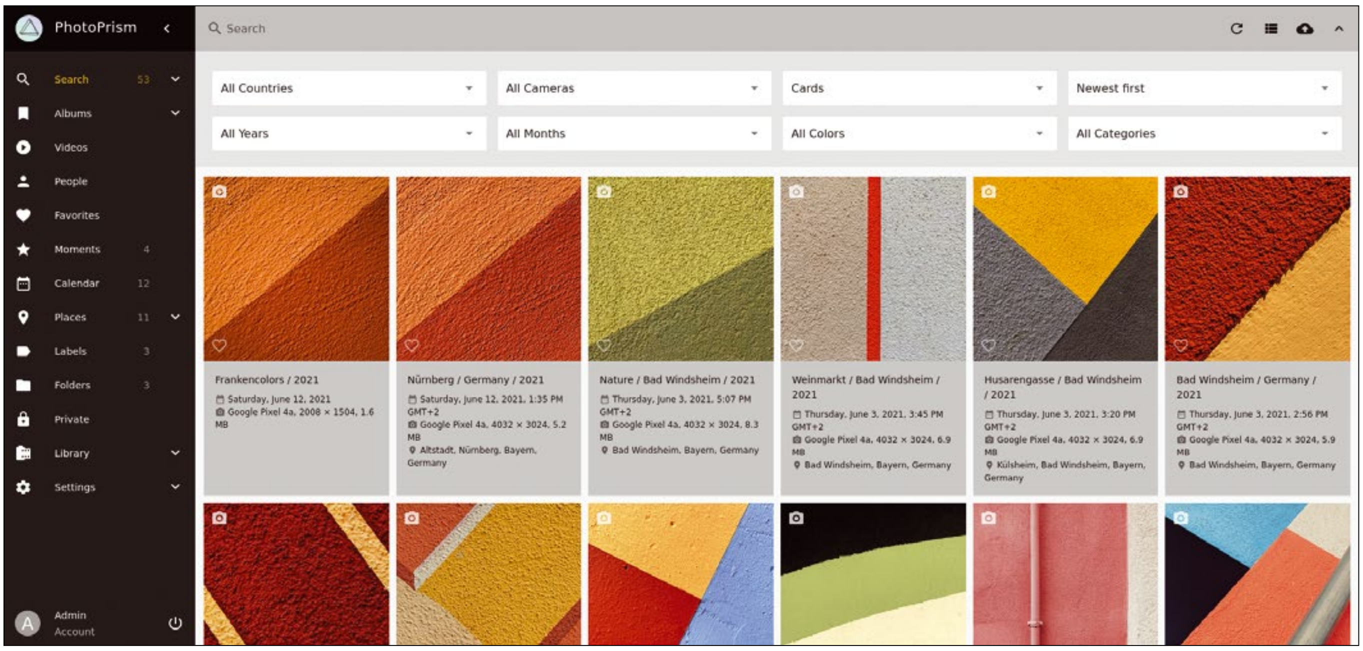

PhotoPrism
Search
Search 53
Albums
Videos
People
Favorites
Moments 4
Calendar 12
Places 11
Labels 3
Folders 3
Private
Library
Settings
Admin Account
All Countries
All Cameras
Cards
Newest first
All Years
All Months
All Colors
All Categories
Frankencolors / 2021
Saturday, June 12, 2021
Google Pixel 4a, 2008 × 1504, 1.6 MB
Nürnberg / Germany / 2021
Saturday, June 12, 2021, 1:35 PM GMT+2
Google Pixel 4a, 4032 × 3024, 5.2 MB
Altstadt, Nürnberg, Bayern, Germany
Nature / Bad Windsheim / 2021
Thursday, June 3, 2021, 5:07 PM GMT+2
Google Pixel 4a, 4032 × 3024, 8.3 MB
Bad Windsheim, Bayern, Germany
Weinmarkt / Bad Windsheim / 2021
Thursday, June 3, 2021, 3:45 PM GMT+2
Google Pixel 4a, 4032 × 3024, 6.9 MB
Bad Windsheim, Bayern, Germany
Husarengasse / Bad Windsheim / 2021
Thursday, June 3, 2021, 3:20 PM GMT+2
Google Pixel 4a, 4032 × 3024, 6.9 MB
Külsheim, Bad Windsheim, Bayern, Germany
Bad Windsheim / Germany / 2021
Thursday, June 3, 2021, 2:56 PM GMT+2
Google Pixel 4a, 4032 × 3024, 5.9 MB
Bad Windsheim, Bayern, Germany

Figure 5: PhotoPrism offers a wide range of filters to refine your search.

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 *https://www.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*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

CloudFest 2022

Date: March 22-24, 2022

Location: Europa-Park, Germany

Website: *https://reg.cloudfest.com/?code=CF22Linux*

CloudFest is the world's #1 cloud computing event where you'll find your next partner in hyperscaling, innovation, and business growth. Take over an amusement park for unforgettable social events, mind-blowing sessions, and the most valuable networking in the industry. Register free with code CF22Linux!

DrupalCon Portland

Date: April 25-28, 2022

Location: Portland, Oregon

Website: *https://events.drupal.org/portland2022*

Come to the Drupal community's largest annual event – in person! Build your skills, learn more about what Drupal can do, and contribute to advancing the open source platform that organizations around the world rely on every day. This year's in-person event will be full of valuable insights, information, and connections.

Events

SCaLE 19x	March 3-6	Pasadena, California	https://www.socallinuxexpo.org/scale/19x
Open Networking & Edge Summit Europe 2022	March 8-9	Antwerp, Belgium	https://events.linuxfoundation.org/about/calendar/
ASWF Open Source Forum 2022	March 10	Los Angeles, CA + Virtual	https://events.linuxfoundation.org/
CloudFest 2022	March 22-24	Europa-Park, Germany	https://registration.cloudfest.com/registration?code=CFMEDIA22
Storage Developer Conference	April 5	Virtual Conference	https://www.snia.org/events/sdcemea
Cephalocon 2022	April 5-7	Portland, Oregon + Virtual	https://events.linuxfoundation.org/cephalocon/register/
ODSC East 2022	April 19-21	Boston, MA + Virtual	https://odsc.com/boston/
LinuxFest Northwest 2022	April 22-24	Virtual Event	https://lfnw.org/conferences/2022
DrupalCon Portland 2022	April 25-28	Portland, Oregon	https://events.drupal.org/portland2022
Linux Storage, Filesystem, MM & BPF Summit	May 2-4	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
Cloud Expo Europe Frankfurt	May 11-12	Frankfurt, Germany	https://www.cloudexpoeurope.de/en
The Open Source, Infrastructure Conference	May16-17	Berlin, Germany	https://stackconf.eu/
KubeCon + CloudNativeCon Europe 2022	May 16-20	Valencia, Spain	https://events.linuxfoundation.org/
ISC High Performance 2022	May 29-June 2	Hamburg, Germany	https://www.isc-hpc.com/
OpenJS World 2022	June 6-10	Austin, Texas	https://events.linuxfoundation.org/openjs-world/
cdCon	June 7-8	Austin, Texas + Virtual	https://events.linuxfoundation.org/cdcon/

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*.

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.

Authors

Mats Tage Axelsson	24
Erik Bärwaldt	30
Zack Brown	11
Bruce Byfield	6, 20, 36, 50
Joe Casad	3
Mark Crutch	75
Andrew Fingerhut	48
Karsten Günther	14
Jon "maddog" Hall	76
Swen Hopfe	70
Charly Kühnast	40
Eva-Katharina Kunst	42
Christoph Langner	77
Vincent Mealing	75
Pete Metcalfe	64
Graham Morrison	84
Dmitri Popov	90
Jürgen Quade	42
Mike Schilli	54
Deepa Seshadri	48
Ferdinand Thommes	60, 80
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com
Copy Editors
Amy Pettle, Aubrey Vaughn
News Editor
Jack Wallen
Editor Emerita Nomadica
Rita L Sooby
Managing Editor
Lori White
Localization & Translation
Ian Travis
Layout
Dena Friesen, Lori White
Cover Design
Dena Friesen
Cover Images
© monsit jangariyawong, 123RF.com and Ken Cheung on unsplash.com
Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420
Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA
Publisher
Brian Osborn
Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

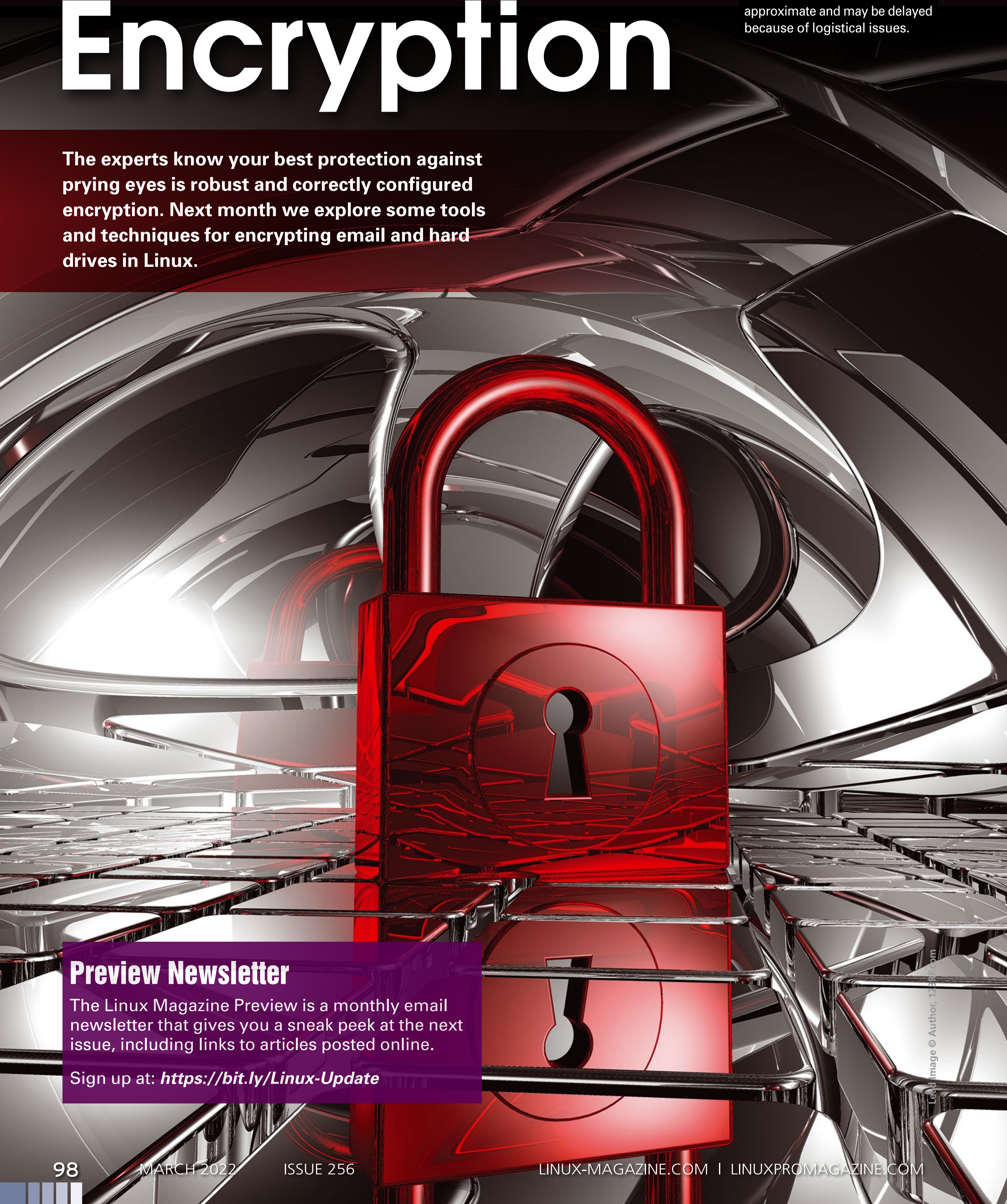
www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.



Approximate

UK / Europe	Mar 05
USA / Canada	Apr 01
Australia	May 02

On Sale Date

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 257 / April 2022

Encryption

The experts know your best protection against prying eyes is robust and correctly configured encryption. Next month we explore some tools and techniques for encrypting email and hard drives in Linux.

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: ***https://bit.ly/Linux-Update***

CLOUDFEST

CONNECTING THE GLOBAL CLOUD INDUSTRY.

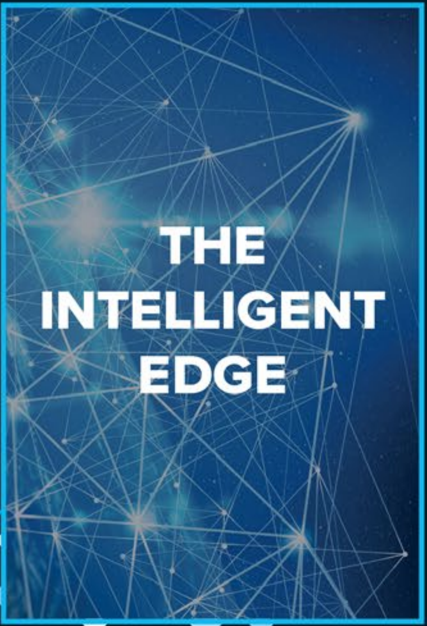
March 22 - 24, 2022 | Europa-Park, Germany

REGISTER NOW AND SAVE € 399!

With Free Code: WE-LOVE-LINUX

REG.CLOUDFEST.COM

CLOUDFEST THEMES FOR 2022



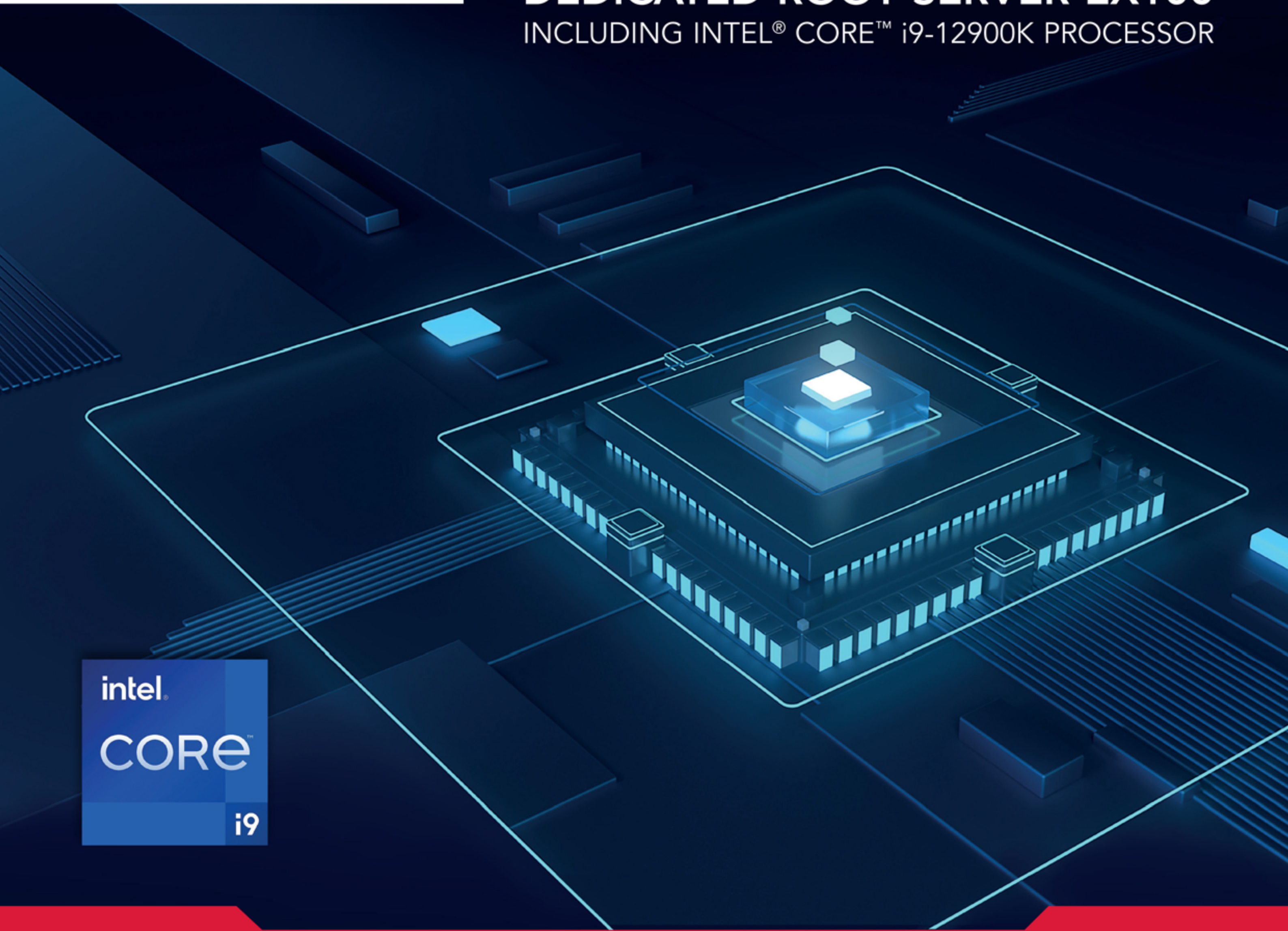
www.cloudfest.com

HETZNER

NEW!

DEDICATED ROOT SERVER EX100

INCLUDING INTEL® CORE™ i9-12900K PROCESSOR



CPU PERFORMANCE REDEFINED

Equipped with the Intel® Core™ i9-12900K processor from Intel's new "Alder Lake-S" generation, the EX100 makes a big impression with its high powered performance and energy efficient CPU cores.

The EX100 will power through workloads in a wide variety of use cases with its 128 GB of DDR4 RAM and two 1.92 TB Gen4 NVMe SSDs Datacenter Edition.

DEDICATED ROOT SERVER EX100

- ✓ Intel® Core™ i9-12900K 16-Core
- ✓ 128 GB DDR4 RAM
- ✓ 2 x 1.92 TB Gen4 NVMe SSD Datacenter Edition
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany
- ✓ No minimum contract
- ✓ Setup fee $125

monthly $ 125

www.hetzner.com