Control your Android from the Linux desktop

LINUX MAGAZINE

ISSUE 292 – MARCH 2025

What Comes After Git?

Next generation version control with Pijul

Mailbot: Build a Python script that captures email addresses from websites

AI on the Pi

Custom Malware Search: Shut down the miscreants by rolling your own rulesets

RHEL AI: Red Hat's Jesper Rooth spills on what's ahead

Linux on Steam Deck: Run your favorite Linux on the popular gaming console

10 ILLUMINATING FOSS GEMS!

LINUX NEW MEDIA
The Pulse of Open Source

FACE TO FACE

Dear Reader,

The business universe changed a little during the pandemic, and the fact is, it hasn't fully changed back. We're in unfamiliar spaces – the working world is clearly not like it was in the past, but did we actually choose these spaces we're in, or are we just here by default? Business leaders are considering these questions now as they contemplate a return from the work-from-home policies initiated during the pandemic years.

According to a recent report in Business Insider [1], Amazon's sweeping return to office mandate didn't go as planned. Employees reported full parking lots and not enough desks, with employees working out of lunchrooms and needing to tie up meeting spaces for private phone calls. It is almost as if they had never done a head count to see how many desks, meeting rooms, and parking spaces they would actually need. And, because a large number of the conversations were with clients and suppliers who were off-site, many of these meetings were still virtual and remote despite the employee's presence in the office.

Amazon CEO Andy Jassy announced the return to office policy last fall [2], stating "We want to operate like the world's largest startup. That means having a passion for constantly inventing for customers, strong urgency (for most big opportunities, it's a race!), high ownership, fast decision-making, scrappiness and frugality, deeply-connected collaboration (you need to be joined at the hip with your teammates when inventing and solving hard problems), and a shared commitment to each other."

I'm always interested in how much the mythology of the "startup" permeates the culture of our business community. What does it mean to make your company "operate like a startup?" I'm sure the intent is for the business to happen in a climate of excitement and youth-like optimism, with deep devotion to the team and a mystical vision of conquering the world. The myth of the startup persists because we only study the successes. The fact is, most startups crash and burn, with almost half going down in the first two years and up to 90% failing eventually [3]. Some burn through the investors' money with remarkable haste. Yet the archetypal vision of the lean and hungry startup has a powerful hold on shareholders and people who study business.

Today's corporations sometimes have millions of employees and billions of interconnected parts, but the CEOs still have to think about them like they are moving chess pieces on a chess board. Amazon's return-to-the-office order affects 350,000 employees. For some, I imagine it really would help to be joined at the hip "when inventing and solving hard problems," but I can't believe that is true for all 350,000 of them.

All this made me wonder what the actual research tells us about the productivity of home versus office work. As you can imagine, the past few years have seen a number of studies on this topic. Results vary, but there is some evidence that fully remote workers are 10 percent or, in some cases, up to 20 percent less productive than office workers. However, a hybrid policy – working some days from home and some from the office, which Amazon had before this change – is thought to deliver overall productivity equal to office work [4].

Of course, every company is different. It all depends on the industry and on the team. My industry has been largely virtual for years. I can certainly see that in-person attendance can be important in a mentoring context, when a new employee is learning the job. But by the same measure, once the job is learned, too much oversight by the supervisor can lead to a culture of control and dependence, which is quite the opposite of a lean and streamlined business environment. It also depends on what you call productivity. If your employees are 10 percent less "productive" but you don't have to rent office space or pay utilities, you could easily come out ahead.

Forcing employees back to an office that isn't ready to receive them looks bad, although it is common for big changes to break things temporarily. I imagine Amazon will work out the kinks and life will return to very much like it was before the work from home era began. Some will leave, but the company is contemplating staff reductions anyway, so that shouldn't stop the music.

The question is, will this change actually put Amazon in a stronger position, or is it largely an exercise designed to placate the board and the shareholders? Only time will tell us the answer – or maybe we won't even get an answer. For today, I'm just a little amused that, when you tell a group of business executives you want to have "more meetings," they frown with grave concern, but when you say you want "more collaboration," their eyes light up enthusiastically.

Joe Casad,
Editor in Chief

Info

[1] "Amazon's Full RTO Is Off to a Bumpy Start": *https://www.businessinsider.com/amazon-rto-issues-space-security-productivity-2025-1*

[2] "Message from CEO Andy Jassy: Strengthening Our Culture and Teams": *https://www.aboutamazon.com/news/company-news/ceo-andy-jassy-latest-update-on-amazon-return-to-office-manager-team-ratio*

[3] "Startup Failure Rate Statistics (2024)": *https://explodingtopics.com/blog/startup-failure-stats*

[4] "Remote Workers Actually Aren't More Productive": *https://www.latimes.com/business/story/2024-01-04/2024-year-employers-clamp-down-on-remote-work-not-so-fast*

LINUX MAGAZINE
MARCH 2025

ON THE COVER

NEWS

COVER STORY

REVIEW

IN-DEPTH

16 What Comes After Git?

Git is practically part of the woodwork in open source circles, but can we do better? The Pijul developers think we can. This month we look at the ambitious Pijul project and the effort to build a next-generation version management system based on patches rather than snapshots.

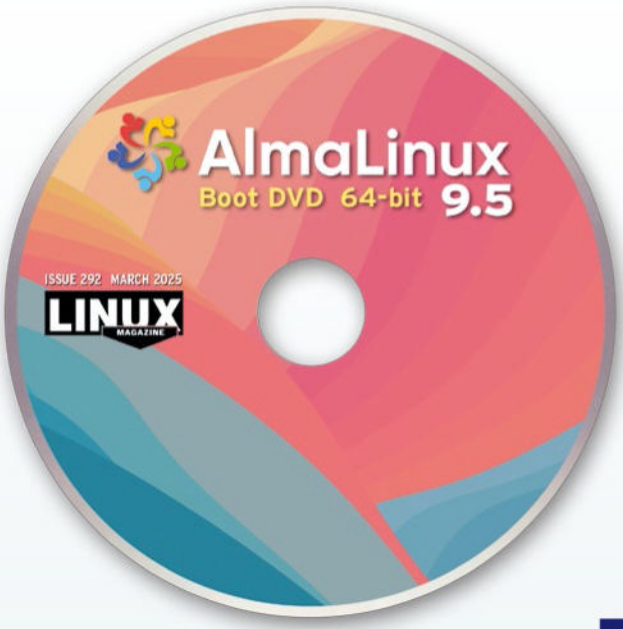
MakerSpace

LINUXVOICE

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

AlmaLinux 9.5 and Manjaro Gnome 24.2

Two Terrific Distros on a Double-Sided DVD!



AlmaLinux 9.5
64-bit

AlmaLinux is a leading community-based alternative to Red Hat Enterprise Linux (RHEL). Its releases are binary-compatible with RHEL rather than sharing the same code. Developed independently of RHEL, AlmaLinux includes hardware support that is not included in RHEL. Originally developed as a replacement for the discontinued CentOS, AlmaLinux's version numbers continue CentOS's numbering system.

AlmaLinux 9.5 adds .NET 9.0 and BIND 9.18 to its collection of toolkits. However, like most point releases, the 9.5 release is mostly concerned with updates of existing packages. These include numerous toolkits, application streams, performance tools and debuggers, and performance monitors. Also featured in the 9.5 release are the usual security updates and device monitors not included in RHEL. For a complete list of these changes, see *https://wiki.almalinux.org/release-notes/9.5.html#changelog*.

Manjaro Gnome 24.2
64-bit

Manjaro is a major Arch Linux derivative that is popular for its installer and its rolling releases, in which new packages from Arch receive extra testing before being made available. Periodically, snapshots of Manjaro are released as images for fresh installs, with different flavors featuring different default desktops.

The Manjaro Gnome 24.2 flavor is built with Gnome 47, which was released in September 2024. Manjaro Gnome 24.2 features include enlarged icons at low resolutions and persistent remote logins. A major improvement is searches in Files: After a search, information is provided about any possible factors, such as remote or unindexed directories, that could affect the quality of results. This information can be used both to refine the current search and to adjust general search settings for better results in the future.

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

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Plasma 6.3 Ready for Public Beta Testing

Plasma 6.3 is ready for beta testing, and it has plenty to offer by way of new features and improvements over current offerings. According to the official blog post (*https://blogs.kde.org/2025/01/11/this-week-in-plasma-final-plasma-6.3-features/*) from the KDE team, there are updates to Bluedevil, Breeze, Breeze-gtk, Discover, DrKonqi, Flatpak permissions, and much more.

After picking through the list, some of the more exciting changes include a new UI for cloning panels that could make it considerably easier to create the exact desktop UI you need; improved Do Not Disturb functionality; a new desktop context menu entry, Show Target; more consistency with close buttons throughout Plasma; an improved look for desktop widgets (with them now being slightly translucent to give them a more modern look); and plenty of bug fixes.

Another interesting change is found within Plasma Discover (the front end for the package manager). Discover will now display for users when apps are packaged directly by the developers or have been verified by a third party. Discover will also highlight any sandboxed apps that have had permissions altered during an update.

At the time of writing, the final release of Plasma 6.3 is expected to drop in February 2025. Remember that beta software (especially a desktop environment) should not be used for production systems.

Budgie 10.10 Scheduled for Q1 2025 with a Surprising Desktop Update

Joshua Strobl, primary developer for the Budgie desktop, recently shared his usual "State of the Budgie" that included goals for 2025 (*https://buddiesofbudgie.org/blog/state-of-the-budgie-2024*). The meat of the piece was all about what happened with Budgie in 2024 (which is interesting on its own), but if you scroll down far enough, you'll get a glimpse into what's planned for 2025.

The first bit of news is the release of version 10.10, which (at the time of writing) is expected in Q1 (between January and March). The biggest announcement, however, is that, with the release of version 10.10, Budgie will be a Wayland-only desktop. Strobl mentioned that in git, Budgie has been Wayland-only since July 2024, so we guess this means standard releases will no longer support X11. This is big news because X11 should be considered no longer a viable option (what with security and performance issues).

At the same time, Budgie 10.0 will be set to maintenance mode to keep it working until Budgie 11 is released.

Strobl also mentioned some of the applets have to be updated to reach some degree of (or complete) feature parity under the "legacy" X11 environments. Those applets include Keyboard Layout, Night Light, Task List, and Workspaces.

Once version 10.10 has been completed, the team will shift their focus to Budgie 11. On that, Strobl had this to say, "After the dust has settled with Budgie 10.10, we will shift our focus towards Budgie 11. While we have already been working on some aspects of the experience that will already apply to Budgie 11, we have not yet started on 'budgie-desktop' itself such as: porting some daemon v1 functionality to v2, windowing logic, panel, Raven, etc."

Serpent OS Arrives with a New Alpha Release

Serpent OS, a community-driven operating system, is a stateless take on Linux that includes atomic updates and a very modern look and feel.

Serpent OS supports NVIDIA GPUs (using the open source kernel modules), can run Steam, includes several Rust-based packages, and offers two different desktop versions: Gnome and COSMIC.

According to the official Serpent OS blog, "Virtually five years in the making, we recently attained alpha status. Our tooling and concepts have aligned, allowing us to now rapidly iterate on the core deliverable itself: Serpent OS." The statement continues, "We've seen an explosion in cadence, with our tooling enabling us to quickly and easily deliver updates, new packages, and enabling new features."

The short-term goals of Serpent OS include offline rollbacks, versioned repositories, improved documentation, enabling IPC in Moss, and improvements in Lichen (the installer).

The latest alpha version includes several fixes and improvements, such as support for Gnome fractional scaling, a pre-built icon theme cache, fixes for AMDGPU initialization, and much more.

You can read the official statement (*https://www.phoronix.com/news/Serpent-OS-Alpha-Update*) and download the latest alpha from the Serpent OS site (*https://serpentos.com/download/*).

Firefox 134 Offers Improvements for Linux Version

As reported first by 9to5Linux (*https://9to5linux.com/mozilla-firefox-134-is-out-with-support-for-touchpad-hold-gestures-on-linux*), the official build of Mozilla Firefox 134 is now available to download from the Firefox FTP site (*https://ftp.mozilla.org/pub/firefox/releases/134.0/*). Firefox 134 includes some important updates, the most user-facing of which is support for touchpad hold gestures.

If you've never used hold gestures, they essentially allow you to interrupt kinetic scrolling by placing two fingers on the touchpad. The best way to describe kinetic scrolling is the movement after your finger has lifted from the touchpad (or screen). The speed of your finger movement defines the duration, speed, and deceleration of the additional movement (after you've lifted your finger). With Firefox 134, you can now stop kinetic movement by simply placing two fingers on the touchpad. This can help prevent scrolling beyond where you want to go on any given page.

Also, with version 134, Firefox now follows the model HTML specification for transient user activation, which makes pop-up blocking less strict (whereas previous versions of Firefox were more aggressive), thereby reducing unnecessary pop-up blocking prompts.

If you're interested in installing Firefox 134, you can download the 32- or 64-bit version (*https://www.mozilla.org/en-US/firefox/download/thanks/*). For a Debian-based distribution, you can set up an Apt repository (*https://support.mozilla.org/en-US/kb/install-firefox-linux*).

At the time of writing, Mozilla has yet to make public the full release notes for version 134, but they should arrive soon.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

SC24 – Bursting at the Seams
• Jeff Layton
We'll recap some noticeable and not so noticeable points of interest that came up during the largest, most attended Supercomputing Conference yet.

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

Optimizing Domain Controller Security
• Thomas Joos
Configure your domain controller security settings correctly with Policy Analyzer and current Microsoft baselines for a leak-tight Active Directory.

Improved Logging in Samba Winbind
• Thorsten Scherf
In Winbind v4.17, the Samba team has addressed the complexity of and difficulty in troubleshooting the logging service that allows Linux systems to join an Active Directory domain.

Manage Status Messages in CouchDB
• Oliver Kurowski
CouchDB offers numerous interesting features for acquisition and filtering of status messages that make it a fast and convenient data storage solution.

Moving from Atlassian Confluence to BlueSpice
• Markus Feilner
With Atlassian's announcement that they are moving to a cloud-only solution, many organizations will be looking for an alternative that lets them keep their collaboration data local. BlueSpice is a worthy open source alternative with easy-to-use migration tools.

HashiCorp Cofounder Unveils Ghostty, a Linux Terminal App

You probably didn't realize that you needed a new terminal app. Or maybe you don't, but you're curious anyway. And given how central the terminal is to Linux intermediate and power users, having the right one can really make a difference.

Mitchell Hashimoto (the co-founder of HashiCorp) understands this and set out to create a new Linux terminal app. This new app is called Ghostty and was written in Zig. The Linux version uses GTK4/libadwaita and the macOS version is written in Swift and uses AppKit and SwiftUI.

The first stable release of Ghostty includes features such as tabs, multiple windows, panes, GPU-accelerated rendering, theming, standard keyboard shortcuts, working directory reporting, programmatic italics, xterm compatibility, custom shader support, ligature and variable font support, grapheme clustering, Kitty graphics protocol, and zero configuration to start using the app.

Ghostty can be installed on Arch Linux (from the Extra packages), but for those using Ubuntu-based distributions, it has to be compiled from source. Hopefully, that will change soon and Hashimoto will release either an official `.deb`, `.rpm`, Flatpak, or Snap package.

You can download the source from the official Ghostty GitHub repository: *https://github.com/ghostty-org/ghostty/releases/tag/v1.0.0*.

Fedora Asahi Remix 41 Available for Apple Silicon

The Fedora Project has released version 41 of the Asahi Remix, which is available for all M1 and M2 series MacBook, Mac Mini, Mac Studio, and iMac devices. This release brings out-of-the-box support for high-resolution audio, a custom setup wizard, and plenty of updated software.

In terms of available desktops, Asahi Remix ships with KDE Plasma v6.2 as the default, but you can opt for Gnome (version 47). Gamers will be thrilled to know that x86/x86-64 emulation has been added with support for AAA games (with the new Vulkan 1.4 driver).

The display, keyboard (with backlight), trackpad, headset jack, speakers, camera, MagSafe (M2 models only), USB-C, WiFi, and Bluetooth features all work out of the box. However, USB-C displays, Thunderbolt/USB4, microphone, and Touch ID currently do not work.

If you're interested in reading more, check out the official site (*https://asahilinux.org/fedora/#device-support*). To install Fedora Asahi Remix 41, you just need to run the following command from the macOS terminal app: *curl https://alx.sh | sh*.

Systemd Fixes Bug While Facing New Challenger in GNU Shepherd

The version of systemd that was released back in June included a bug that could lead to everything in `/home` being deleted if the systemd command is run incorrectly to delete temporary files. This happened because `systemd-tmpfiles` (originally created to manage temporary files) evolved into something much *bigger*. Well, that bigger something wound up causing a pretty serious issue.

To fix this issue (*https://www.msn.com/en-us/news/technology/systemd-begrudgingly-drops-a-safety-net-while-a-challenger-appears-gnu-shepherd-10/ar-AA1vNMPi*), the systemd developers made a change to the format of a config file (that isn't backward-compatible) making it less likely `/home` will be deleted. It's not an ideal fix, but it works.

The systemd v256.1 contains the fix (which should be installed via a standard update). If you're running version 256.0, no patch has been applied, and you should be careful when running the `systemd-tmpfiles --purge` command.

At the same time, a new challenger, GNU Shepherd, has arrived to give systemd a run for its money Okay, GNU Shepherd isn't actually new, as it was first developed over 20 years ago. What's important is that it has finally, after 20 years, reached its first stabile release (version 1.0). The biggest difference between GNU Shepherd and systemd is that GNU Shepherd is developed with Guile Scheme and serves as the default init system for the GNU Guix distribution.

There's little to no chance that GNU Shepherd will be replacing systemd anytime soon, but at least there's more competition in the market, which always leads to more innovation.

AlmaLinux 10.0 Beta Released

AlmaLinux 10.0 Beta offers plenty of changes for development, security, and performance workflows. You'll find updated programming languages, toolchains, and compilers, as well as control systems, servers, and databases.

One of the more fascinating additions is support for post-quantum cryptography, which helps to secure AlmaLinux against cryptanalytic attacks by a quantum computer. Additionally, SELinux and OpenSSH have both been updated, and a new sudo system role makes configuration management across multiple systems easier. Finally, Sequoia PGP has been added for expanded encryption options.

Not to be mistaken for AlmaLinux Kitten (which is based on CentOS Stream and ships with even newer versions of software), AlmaLinux 10.0 Beta is based on RHEL 10.0.

According to the AlmaLinux press release, "Beta releases continue to provide our community with early access to new features and enhancements, allowing test[ing] and encouraging our community to provide valuable feedback before the stable launch."

It's important to remember that this is a beta release candidate, which means it should not be used for production machines or workflows.

AlmaLinux 10.0 Beta is available for the following architectures: Intel/AMD (x86_64), Intel/AMD (x86_64_v2), ARM64 (aarch64), IBM PowerPC (ppc64le), and IBM Z (s390x).

If you're interested in testing this beta release of AlmaLinux 10.0, you can download an ISO from the official repo: *https://repo.almalinux.org/almalinux/10.0-beta/isos/*.

Gnome 47.2 Now Available

As we've all come to expect, it's a rare occasion that a point release introduces new features, and Gnome 47.2 is no exception. This latest release is all about fixing issues that have popped up in the weeks since Gnome 47.1 was made available.

Some of the notable fixes include touchscreen drag and drop on Wayland, as well as a few cursor issues that have arisen (particularly when using a virtual monitor); the accessibility of the keyboard backlight toggle; CPU stalls with NVIDIA GPUs that have directly attached monitors; and a real-time priority default for KMS threads.

With partially rounded buttons, an artifact issue occurred in the previous release, which has also been fixed. Issues with PackageKit and Snap apps have also been resolved, as well as a DuckDuckGo issue in the Epiphany browser that led to redirect failures and a problem when the browser attempted to load GlobalProtect URLs.

A handful of apps have received some attention, such as Orca Screen Reader, Gnome Text Editor, Gnome Maps, Loupe, and Totem. You can read the list of all the modules that have been updated here (*https://download.gnome.org/core/47/47.2/NEWS*).

Keep in mind that you'll want to hold off until Gnome 47.2 arrives in your distribution's repositories, which could take some time depending on your distro of choice.

Zack's Kernel News

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

By Zack Brown

There Are Standards and Standards

One of the most bizarre areas of kernel development lies in the relationship between the kernel and the compiler. Since the creation of Linux at the start of the 1990s, there have been wars fought between the developers of these two projects. In one such case, Linus Torvalds refused to recognize any GNU C Compiler (GCC) version after a certain date, on the grounds that later versions produced bad machine code.

Relationships have become a lot less strained between the two projects since then. However, the question of mending fences does sometimes arise. And it's not always clear whether a problem in the Linux kernel would be best solved by changing the kernel source code or changing the compiler source code. Sometimes the best answer may be simply whatever avoids war.

Recently, Vincent Mailhol tried to simplify some kernel code, relying on the features of the C11 standard for the C programming language that Linux is written in.

The latest official C language standard is actually not C11 but C23, which was released by the International Organization for Standardization (ISO) in October 2024. The C11 standard was originally released in 2011, but Linus takes a very conservative approach when it comes to standards adoption. He updated the kernel's officially supported C standard to C11 with the release of Linux v5.18, which he released in May 2022, 11 long years after the ratification of C11.

Vincent, taking advantage of this relatively recent upgrade to the supported standard, felt that he could make a gentle improvement to the following piece of abject insanity:

```
#define __is_constexpr(x) ➥
  (sizeof(int) == sizeof(*(➥
  8 ? ((void *)((long)(x) * 0l)) : ➥
  (int *)8)))
```

He described it in his email as "one of the most glorious one liner hack[s] ever witnessed by mankind."

That bit of code defines the `__is_constexpr` macro, which tells whether `x` is a constant (in other words, something evaluated at compile time) or not (in other words, something calculated while the kernel is actually running). This is important! If something is a constant, then maybe it can be optimized into oblivion at compile time and thus give the kernel a little more speed. If it's calculated at runtime, well, you just have to live with however long it takes to calculate it.

So how does this macro figure out if `x` is a constant or not? Well, pretty much the whole point of this macro is to answer the question without actually having to do much calculation and without breaking anything. So the calculation basically tries to check the type of `x`, which can be attempted without actually doing the work of evaluating `x`. If `x` is constant, its type is known, so the macro succeeds (i.e., it comes back as `TRUE`). But if `x` is calculated at runtime, its type isn't known yet, so there's a bit of a cascade of harmless failures, resulting in the macro itself failing (i.e., it comes back as `FALSE`). All the insanity surrounding `x` in that macro is just to make sure the cascade of failures is harmless and yet can still be seen.

Certain twisted souls such as Vincent can not only understand what that macro is doing, but they can also be inspired by slight differences between one C standard and the next to find ways to simplify that macro, deriving a satisfaction of a deep and profound kind. Vincent started off by saying, "Following the adoption of C11 in the kernel, this macro can be simplified," essentially by getting rid of the `sizeof` parts of that macro.

As a bit of historical trivia, Vincent also pointed out that the `__is_constexpr` macro "relied on the GNU extension that sizeof(void) is 1." He said that with his rewrite, the new macro (or macros, since he intended to split the original into

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**.

two) would, in addition to the `sizeof` improvement, also eliminate that reliance on the GNU extension and thus bring it into true ISO compliance.

The new code – which Vincent readily affirmed was not entirely his own design – was itself again a bit of abject insanity, although improved:

```
#define __is_const_zero(x) _Generic(
  0 ? (void *)(long)(x) :
  (char *)0, char *: 1, void *: 0)
#define is_const(x) __is_const_zero(
  0 * (x))
```

Obviously the above is a vast improvement and a great simplification over what had been there before, said no one ever, in any universe.

But essentially, the `_Generic` construct, introduced in the C11 standard, arranges a conditional based on the type of what it's looking at. This means you can do away with the `sizeof` shenanigans that accomplished the same thing but that relied on their quirks and oddities rather than their officially intended purposes.

I personally find it funny that the original code relied on using the value of `8` in a conditional in order to always evaluate as `TRUE` and thus never execute one whole branch of its conditional, while the new code relies on using the value of `0` in exactly the same way, but this time to always evaluate to `FALSE`.

There was no serious controversy about Vincent's patch, but David Laight pointed out that the patch submission itself should maybe give credit to the various other developers who had worked on similar obfuscations – I mean optimizations. He listed Linus as one such developer. Vincent offered to list Linus as the person who first suggested this new form of the macro. However, at this point Linus himself came into the discussion, remarking, "I'm generally the one person who doesn't need any more credit ;)."

However, David remarked, "I actually suspect the first patches to change __is_constexpr() to use _Generic were from myself," referencing a patch he had submitted in November of 2023.

Linus confirmed that David was the first to make the attempt. And – although not explicitly explained in the email discussion this time around – I think the `!!` construct must have been bandied about at one time or another as a potential piece of a possible implementation of these macros. The `!!` in C is actually a double negation. It's used as a way to first negate a value and force it into a Boolean form and then to negate that negated Boolean again, to match the truth value that would have corresponded to whatever the original value was. So if you apply `!!` to `7`, it would first negate `7` and force a value of `FALSE` and then negate that negation to get `TRUE`. So, `7` would evaluate to `TRUE`.

Linus said:

"David was also I think the one who suggested something else than '!!' originally too.

"I may have liked '!!' for being very idiomatic and traditional C, but there were those pesky compilers that warn about 'integer in bool context' or whatever the annoying warning was when then doing the 'multiply by zero' to turn a constant expression into a constant zero expression.

"So that

```
#define is_const(x) __is_const_zero(
  0 * (x))
```

"causes issues when 'x' is not an integer expression (think 'is_const(NULL)' or 'is_const(1 == 2)'.

"Side note: I think '(x) == 0' will make sparse unhappy when 'x' is a pointer, because it results in that horrid 'use integer zero as NULL without a cast' thing when the plain zero gets implicitly cast to a pointer. Which is a really nasty and broken C pattern and should never have been silent.

"I think David suggested using ((x)?0:0) at some point. Silly nonsensical and complex expression, but maybe that finally gets rid of all the warnings:

```
#define is_const(x) __is_const_zero(
  (x)?0:0)
```

"might work regardless of the type of 'x'."

Vincent replied, "Then, I will add a suggested-by tag to credit David!"

Believe it or not, there followed a technical discussion in which various folks, including Linus, David, and Vincent, debated alternative implementations and the corresponding implications of those implementations. They were not

satisfied to allow this bit of abject insanity to slip unremarked into the kernel, but needed to proclaim their respective love of the pure insanity of it all.

In response to one such attempt by one of the participants, Linus remarked, "Oh Christ. You really are taking this whole ugly to another level."

In fact, much of the discussion centered around trying to find an implementation that would avoid all compiler warnings. You may remember Linus recently decided that all compiler warnings had to be fixed, and no future warnings would be accepted. It was a tough move, met by wails of anguish from thousands of people across the world. But among other things, it meant these sorts of compiler hacks need a lot more attention and can't just be slapped together the way they used to be.

To give you an example of the seriousness with which these lunatics tackle such problems, at one point in the discussion David offered this implementation:

```
#define const_true(x) _Generic(
  0 ? (void *)((x) + 0 ? 0L : 1L) :
  (char *)0, char *: 1, void *: 0)
#define const_expr(x) _Generic(
  0 ? (void *)((x) + 0 ? 0L : 0L) :
  (char *)0, char *: 1, void *: 0)
```

Saying, "I make that 98 characters. Of course, you can remove all the spaces, only one of the constants need the L suffix and 'int' is a shorter type name. That cuts it down to 76."

However, at one point, Martin Uecker commented with patient sobriety:

"I find it amazing how much time the Linux kernel community spends revising code to make it work perfectly.

"Still, I am wondering whether some of this time and effort should not be targeted at C compilers and language work to make these macro hacks unnecessary?

"I already found the original motivation for these macros very questionable. Removing VLAs [Variable Length Arrays] at the cost of having imprecise worst-case bounds strikes me as fundamentally misguided – at least if security is the motivation.

"So maybe there are other good reasons for this, e.g. bad code for VLAs or risk of jumping the guard page if the attacker can somehow influence its size (but for this there is -Wvla-larger-than). But even then, wouldn't it be a more worthwhile and interesting investment of engineering resources to improve code generation / warnings at the compiler level?"

To which David remarked with sober patience, "I'm probably not alone in thinking that sometimes the compiler writers are doing their hardest to make life hard for people writing low level code."

Vincent also said in response to Martin, "the core issue is that before getting this support in Linux, we have to wait for this to be added to the C2Y draft, then implemented in the compilers (probably just reusing the C++ constexpr functions) and finally wait maybe one more decade for the C2Y support to reach the kernel. For reference the kernel supports C11 only from 2022? So maybe we will see those in the kernel around 2037? Meanwhile, we have to deal with those hacks."

Martin retorted earnestly, "If we do not collaborate on proper solutions, then you might have to wait much longer." But he did also point out, "GCC and Clang are open-source projects just like the kernel. One can go there and contribute. I am not saying that it is always easy to find consensus and there [are] also projects that have other requirements than the kernel. But I started to contribute to GCC (with very limited time) to address some of my own issues, and I found the community very welcoming."

Vincent, by way of indicating that he was very willing to work with the compiler people, replied, "I was invited to WG14 [the C language standards committee meeting] this September. For now, I am only lurking. The thing I have in mind right now is to write a paper to allow the use of static_assert() in expressions (i.e. make it return 0 on success). That should be a relatively small change, but would bring a nice quality of life improvement."

Martin replied in good-natured spirit, "Excellent, then I was complaining to the wrong person."

The discussion continued, wending this way and that way through time and space. At one point, Linus revealed his thinking on supported compiler versions, saying, "We're currently still accepting gcc-5.1 as a compiler, although it's time to look at that and probably (judging by what stable distros use) upgrade to something like gcc-8.1 as the minimum supported compiler version."

To which David replied:

"That's going to annoy me. The system disk in the system I test build [the] kernel on is actually older than the machine! (not by much). And Ubuntu 18.04 (still getting some fixes) has gcc 7.5.0.

"It isn't as though the 8.1 update is anything really major. Disabling stack canaries would let an older compiler be used. (and I might change the tests…)

"Much more useful would be mandating 'asm go with outputs' which would cut out a whole load of horrid alternatives.

"But that would make it pretty common that a kernel build would need a later compiler than the one the distribution installed.

"It may be time to consider directly supporting downloading and building the required compiler as part of a normal kernel build. That would allow the minimum version to be set to a very recent build and also make cross architecture build easier. (In effect all builds become cross builds.)

"NetBSD used to (may still do) import gcc into its CVS repository. So that everything was built with a 'known' compiler.

"It is (probably) less of a problem with clang. People using clang are likely to have explicitly downloaded it."

In fact, there were a lot of tasty nuggets in this conversation, and I very much enjoyed reading it. It's hard to pick and choose what to include in limited space.

However, the main discussion does not seem to have been in any serious way controversial – a complex macro was made slightly simpler, and a bunch of people who spend their days solving really hard problems got to have a bit of fun together on this one. ■■■

DrupalCon
VIENNA2025
14-17 OCTOBER

Mark your Calendar for DrupalCon Vienna 2025!

Get ready to connect, learn, and innovate! From **14 - 17 October 2025, Vienna** hosts **DrupalCon Europe** once again. Whether you're a developer, designer, marketer, or business leader, join us to share ideas, collaborate, and shape the future of Drupal.

Set in the heart of one of Europe's most inspiring cities, DrupalCon Vienna promises a unique blend of history, creativity, and technology. Don't miss out on the opportunity—mark your calendars and **join us for an unforgettable experience!**

Make sure to check the official website and follow us on social media regularly to stay updated and never miss important news!

@DrupalConEur

Exploring the innovative Pijul version management system

The Wild Bird

What comes after Git? The Pijul developers think they know the answer. The Pijul version control system blends old ideas in a smart new way. Will it be enough to kick off a post-Git era?

By Tim Schürmann

The groove-billed ani, which is native to Mexico, is a bird that builds its nests communally in small groups. Although these cute little birds have a tongue twister of a name in English, the Mexicans simply call them Pijuls. Pijul is also the name of a relatively new version control system created by Pierre-Étienne Meunier and Florent Becker [1]. Like the groove-billed ani, the Pijul version control system focuses on distributed teamwork. Pijul is also extremely fast and easy to use, and it does away with some of the problems that haunt Git users. At least that's the full-bodied promise of the makers.

At first glance, working with Pijul actually seems surprisingly simple: After you have registered all files with the version management system by typing `pijul add *`, you communicate every change to the tool by typing `pijul record`. And that's it. Unlike Git, there is no confusing staging area, and the commands are far leaner than Git commands.

Pijul uses the same distributed approach: Each team member has a local repository in which the version management system stores all changes made by the local author. Exchanging data between team members is also a breeze with Pijul. If programmer Alice wants to adopt a bugfix from her colleague Bob, she types `pijul pull` to pull the code change into her own copy. Even if Bob has made more changes in the meantime, Alice can easily cherry pick the bugfix with `pijul pull`; an equivalent to this kind of cherry picking can require many steps in Git.

Weighty Snapshots

Git and many other well-known version control systems regularly take snapshots of the current work in progress. Each of these snapshots builds on the previous one. If you want to restore a snapshot, you need its predecessor, which makes cherry picking more difficult. With Git, you often have to resort to `git rebase` [2] or `git rerere` [3].

A distributed work approach also inevitably leads to conflicts. For example, what if Bob has edited the same files as Alice? Although version control systems try to autonomously merge the content of the files, they sometimes miss the mark, leaving you with scrambled lines of code [4]. Pijul takes a different approach, which the developers have copied from an old acquaintance.

Back in 2003, before Git was invented, David Roundy published the Darcs [5] version control system. Darcs does not rely on snapshots but tracks the individual changes in the files. The repository only stores patches, each of which represents a very specific change to the actual content. This approach has several advantages. First of all, the order in which you apply a couple of changes to your own repository no longer matters. Alice can first retrieve the bugfix from Bob's repo and then the code for a new feature – or vice versa: adopt the new code first and then apply the bugfix. In both cases, the results will be the same (Figure 1). If you think back to your math classes, you'll remember that this property is referred to as commutative.

It also makes no difference whether Alice integrates the two changes into her source code one after the other or whether she first combines the changes and then adopts the results into her repository. The changes are associative (Figure 2). Changes can be adopted step by step. This associative property also prevents the version management system from jumbling up the lines of code during a merge.

Slowcoach

These nice-to-have features included with Pijul make cherry picking and merging radically easier. Because the version

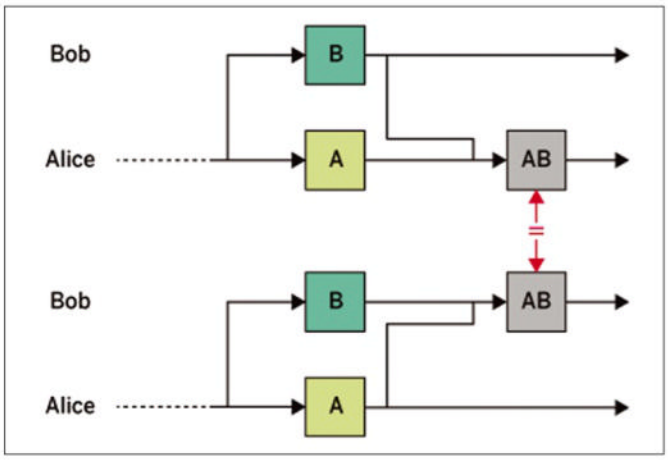


Figure 1: In the case of commutative changes, it does not matter whether Bob accepts the change from Alice or vice versa. The results are identical in both cases.

management system only handles the source code changes, operations are also simplified. Generally speaking, this only works if the individual changes are actually completely independent of each other. And conflicts can still occur, say, if Alice and Bob edit the same line.

Darcs solves these problems with a healthy dose of complex mathematics. The calculations this requires can take several hours, even for smaller code changes. As a result, Darcs is even slower than most counterparts, and on top of that, Darcs scales quite poorly.

Nodes and Edges

Pijul manages to avoid many of the performance and scaling issues associated with Darcs. Under the hood, the changes tracked by Pijul are managed by a sophisticated data structure; Figure 3 shows a simplified example. The starting point is a graph consisting of nodes and edges (the arrows). In this structure, the tool remembers which change modified how many bytes at which point. Although the graph looks pretty confusing, it is quickly explained.

First of all, someone fills an empty file with *i* bytes. Pijul numbers all the changes internally, with the first change being assigned the number 0. Pijul dumps the information into the node on the far left in Figure 3, the change number is *C0*. The *C* stands for change number and is only intended to help differentiate between the nodes in the graphic. The node also indicates positions at which something in the file changed (*0* to *n-1*). The file was opened initially, then *n* bytes were added. Writing started at position *0* and ended at position *n-1*.

In the next step, additional bytes were inserted in the middle of the file *m*. This change is named *C1* and involves the insertion of *m* bytes in total, starting at position *p*. To do this, Pijul splits the file at this position, pastes the new *m* bytes and appends the rest of the original file.

The first bytes from the first change *C0* from position 0 to *p*-1 are kept. These bytes are followed by the *m* bytes from the second change *C1*, and all bytes from position *p* to *n*-1 from *C0* following at the end.

Darling, It's a CRDT

There is also an edge that leads from the modified file back to *C0*, which allows Pijul to reconstruct the original. Deleting data

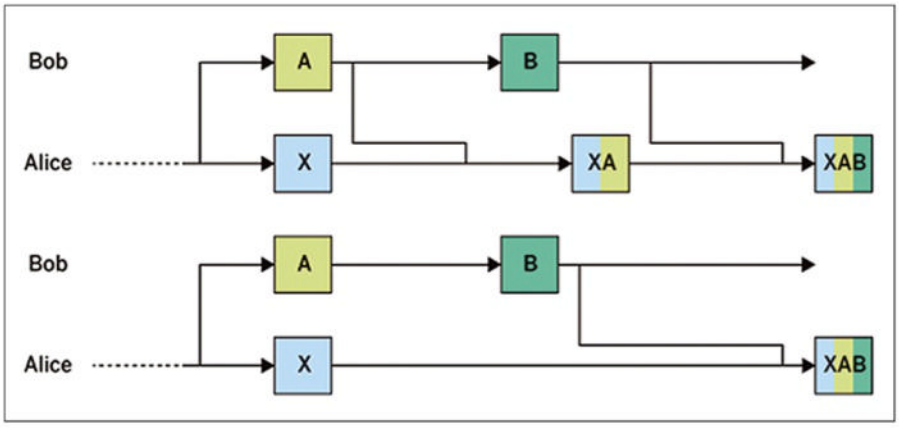


Figure 2: Alice can either adopt associative changes from Bob one after the other (above) or directly together. The results are always the same.

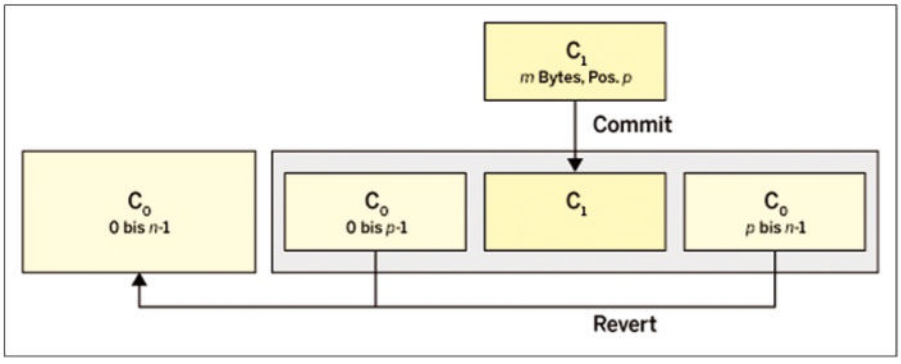


Figure 3: Change C_1 inserted precisely m bytes into the file at position p.

works in a similar way to the insertion process I just described; no further operations are necessary.

Admittedly, the data structure in Pijul is somewhat more complex than shown in Figure 3. Among other things, you can see when which developer modified which lines. Pijul relies on advanced mathematics, for example, category theory. The result is a conflict-free replicated data type (CRDT). Data structures of this kind can transfer to different computers where users can change them independently of their team colleagues – and this is exactly what is needed for distributed version management.

Thanks to its clever architecture, Pijul enables efficient cherry picking and also lets you clone a specific part of a repository into a new one (partial cloning). It is also possible to merge two repositories. Last but not least, you can even undo historic changes.

Internal Phone Directory

The graph in Pijul contains a huge number of tiny bits of information, which means that it grows quickly and no longer fits easily in RAM. The size problem prompted the Pijul developers to find a way to store the graphs as efficiently as possible on a data carrier.

They found a solution in the form of a key-value store, that is, a database that only stores key-value pairs. But none of the existing key-value stores on the market met the requirements for innovative version management. The need for something better drove the Pijul makers to develop their own database: Sanakirja, which means *dictionary* in Finnish. According to the Pijul team's test results [6], Sanakirja is faster than existing alternatives such as Sled [7] and LMDB [8].

Missing Functions

Pijul started out as a kind of experiment and research project, and even today only a small team of developers keeps things moving along. At the time this article went to press, the version management system was still beta, but the first stable version 1.0.0 was already on the home straights. Right now, it is not advisable to entrust critical projects to Pijul, although the software is already quite stable in practice. In fact, the Pijul team relies on it as its own version control system: Pijul is versioning itself.

Currently, Pijul does not provide support for tags that would let you mark beta versions or release candidates. (Git has offered tag support for some time.) Other features are available, but are not well tested.

In some cases, developers want to combine several individual projects in a single large repository. Monorepositories like this have proven their value, for example, with applications that consist of numerous microservices. According to the developers, Pijul scales excellently and is therefore suitable, in principle, for running very large repositories. Having said this, the Pijul developers point out in the FAQ that they have not focused on operating a monorepository to date. In other words, it could work, but there are no guarantees, especially because Pijul does not currently offer any features specifically tailored to monorepositories.

Bigger Things

Although Pijul can process very large files, merging might not be efficient, depending on the data stored in the files. This situation applies to large video files, for example. But, due to the way version management works, you do not necessarily have to download all the editing steps of the files. For example, if a video journalist makes several changes to a movie before submission, the customer does not have to download these intermediate versions from the Pijul repository.

Another problem arises as soon as you make large-scale changes to a complex project. This can easily happen if style guides change and you need to reformat all the functions in all the source code files. Numerous dependencies are then

Pijul and Git

The Pijul developers are quick to criticize some disadvantages of Git, although they also affect Pijul. For example, they complain that existing version management systems are used almost exclusively by developers. But, in order to use Pijul sensibly, you need to understand the underlying concepts. In addition, version management in its current state favors text files. Pijul also has many subcommands, some of them with multiple parameters. This makes life difficult for technically inexperienced users who might only want to version LibreOffice tables in an easy way.

In fact, the claim the Pijul developers make is simply wrong: Git is now often used for managing general texts, such as style guides or Ansible cookbooks. Thanks to libgit2 [9], a Git repository can also be integrated into your own applications and can then hide all details from users. This is how Subsurface [10], for example, stores divers' logbooks transparently in Git repos.

Although Git is a distributed version control system, most projects have a central repository, typically on GitHub or GitLab. The team members will tend to synchronize with this repository and make their changes there. The many different existing platforms are a thorn in the side of the Pijul developers. In their opinion, platforms are currently springing up like mushrooms, and they believe that this is not due to different requirements but to fundamental problems with Git. What this ignores is the fact that users and developers have a choice with Git. If you don't like GitHub, for example, you can easily switch to GitLab. In contrast to this, Pijul users are dependent on the project's own offering.

Anyone who has ever worked with branches in Git knows that they tend to spread in larger projects like branches in a rainforest. Working with branches requires good planning. It seems that the Pijul developers are not enamored of branches either. In their opinion, branches are superfluous in Pijul. They argue that feature branches, in which new software functions are created in isolation, are often nothing but changes. If you simply continuously push all your own changes into the version management system, you can quickly lose track of previous actions. If there are many changes, you also always have to painstakingly search for the changes that belong to a specific feature. The Pijul developers have at least created a compromise in the form of channels, which behave in a similar way to branches.

Dependencies

To install Pijul, you will need Rust, the Rustup and Cargo tools, Make, Clang, Pkg-configuration, and the diffutils. You will also need the Libsodium, Libclang, OpenSSL (in the form of Libssl), Libxxhash, and Zstd libraries (in the form of Libzstd) along with their developer packages.

Listing 1: Installing on Debian/Ubuntu

```
$ sudo apt install make libsodium-dev libclang-dev pkg-config
  libssl-dev libxxhash-dev libzstd-dev clang rustup
$ rustup default stable
$ cargo install pijul --version "~1.0.0-beta"
$ export PATH="$PATH:$HOME/.cargo/bin/"
```

created, which also migrate to the repository. Pijul users need to think ahead and push changes into the version management system as independently as possible. For example, each file could be reformatted individually and then registered as a separate change in Pijul. See the box entitled "Pijul and Git" for more on some of the complications associated with Pijul.

Getting Started

If you are curious about Pijul, you can install it just a few steps. All the components you need are listed in the box entitled "Dependencies."

On Debian and Ubuntu, the command in the first line of Listing 1 installs the stuff you need. The next line ensures that you only use the stable Rust tools. The next line sets up Pijul, and the last line adds the path to the `pijul` program to the `PATH` variable.

If you have worked with a version management system previously, and with Git in particular, you will easily find your way around in Pijul. Start by changing to the project directory with the files you want to version and create a new repository there by typing `pijul init`. The new repository is stored in a new hidden subdirectory named `.pijul/`. The version management system remembers which author made each change. For this to work, you first need to enter some details of your identity after typing `pijul identity new` (Figure 4).

You can call up the command multiple times to create several identities, say, for different projects or teams. In order to distinguish the new identity from the others, you must assign a unique name in the first step. Pijul also asks you for your name, your email address, whether or not you want to password-protect this identity, whether to expire the identity on a specific date, and whether or not to link it to a remote user account. You can say no to the last three questions.

When creating the identity, Pijul generates a cryptographic key (public key) that it uses to sign each of your changes. The key also always uniquely identifies an author and cannot be manipulated as easily as a name or email address.

Next Steps

Once you have stored your identity, you can hand over one or multiple files to Pijul for safekeeping. If you are working on a Go program named `hello.go`, for example, the command would be `pijul add hello.go`. Instead of a single file, you can also specify multiple files or an entire subdirectory. And if you want the tool to ignore files, just enter the filenmes in the `.ignore` text file.

Pijul now knows which files you want it to manage, but the repository is still empty. To fill it, enter your source code in the `hello.go` file and tell Pijul about the change. To record the change, simply call `pijul record`. After you choose a text editor, Pijul opens a view with a mass of information about the change (Figure 5). You can enter a comment in the quotes following `message =`.

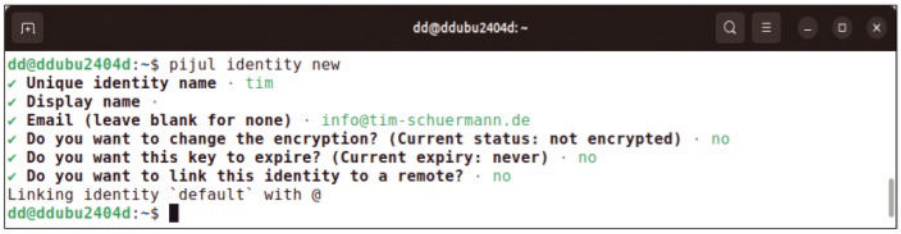


Figure 4: You need an ID in Pijul before you can manage files, otherwise the version management system will refuse to work.

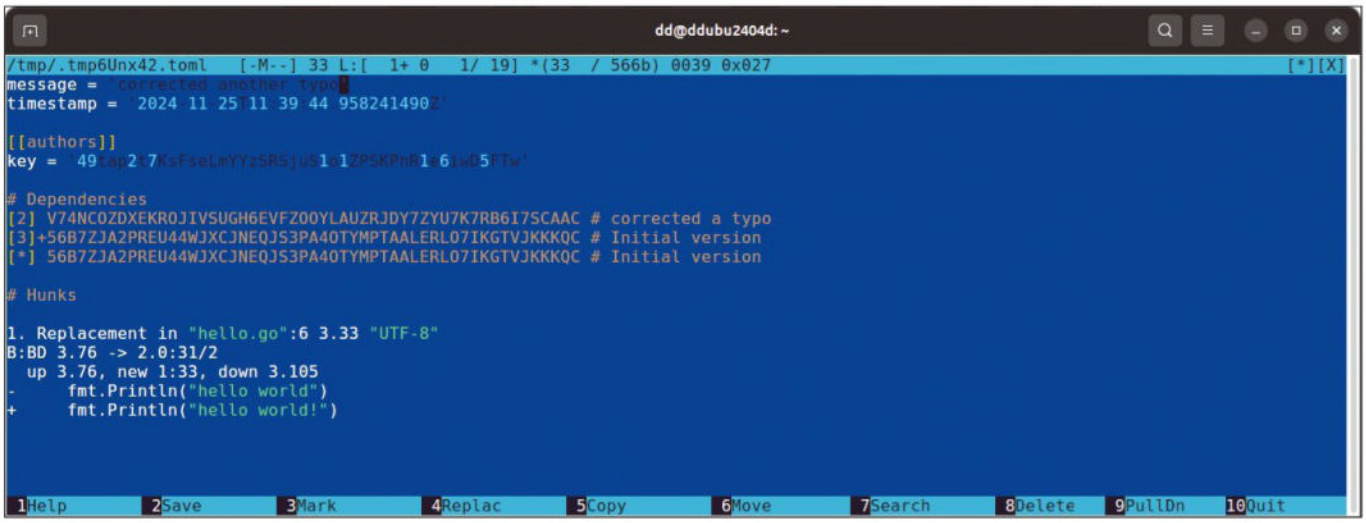


Figure 5: Using the `pijul record` command to view all the key data of a change and correct the data if necessary.

Figure 6: Each change is given a unique and cryptic identification number, which `pijul log` lists after a change.

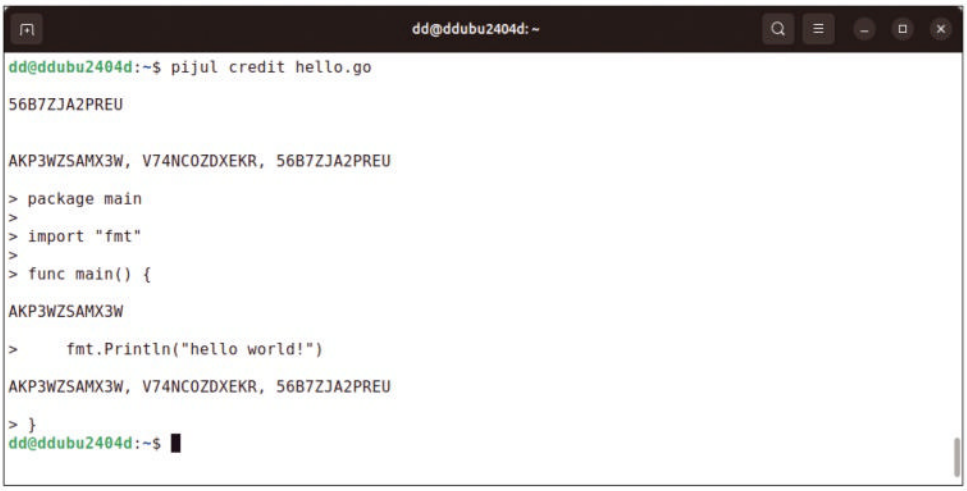


Figure 7: `pijul credit` tells you which developer edited which lines and when.

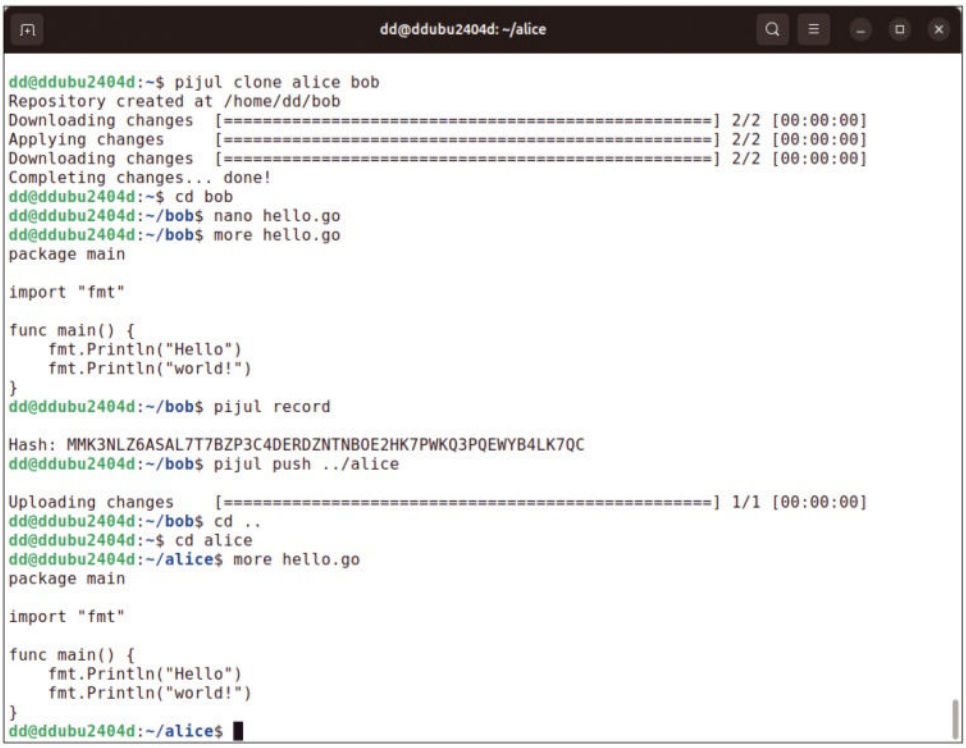


Figure 8: Pijul using `push` to transfer the changes from the `bob/` directory to the repository below `alice/`.

From now on, you need to call `pijul record` after every change to your files. `pijul log` lists all the changes (Figure 6). You can undo a change by typing `pijul unrecord`. This undo feature is particularly useful for resolving conflicts. The command expects the fairly cryptic ID number as a parameter; `pijul log` shows you this parameter after `change`. `pijul credit` (Figure 7) is the more nicely formulated equivalent of `git blame`.

Collaboration

Accessing a second repository works in a similar way to Git. First, run the command `pijul clone /home/tim/helloworld` to clone the repository from `/home/tim/helloworld/` in your current folder. Instead of a local repository, you can also use SSH (Listing 2, first line) or HTTPS (last line) to access a repository.

You can now use `push` to move changes created in one of the two repositories to the other, or conversely, use `pull` to drag them in. Figure 8 shows an example. As you can see, Pijul is cloning the repository that resides in the `alice/` directory in the `bob/` directory. Then the `hello.go` file is edited. `pijul push` pushes this change back into the repository in `alice/`. When you call `push`, a text editor opens again, and you can delete the entries for the changes that you are not transferring.

Channels

After delivering a completed program, the development will normally continue, meaning that changes are continually added. But it is a good thing to know which changes affect the application version in production use.

Pijul offers developers channels to designate different versions of the software; a channel is really no more than a bunch of changes. In practical terms, you first need to create a channel, assign a unique name such as

Listing 2: Accessing Remote Repositories

```
$ pijul clone tim@myserver.de:pathto/repo
$ pijul clone https://myserver.de/pathto/repo
```

Listing 3: Channels

```
$ pijul fork stable
$ pijul record --channel stable
$ pijul channel switch stable
```

`stable` and move a new change (line 2) into the channel (Listing 3). You can save yourself some of the typing by switching directly to the channel with the last command in Listing 3. The commands in Listing 3 assure that all changes are written directly to `stable`. Typing `pijul channel` outputs a list of existing channels.

Pijul channels are similar to Git branches in terms of the way they work. But because a channel consists of a collection of changes, it behaves in a slightly different way. There is no command for merging two channels. Instead, you simply pull the required changes from the channel. To get started, type `pijul log --channel stable`, determine the identification number of the last change, and then apply the ID by typing `pijul apply ABC`, where `ABC` stands for the ID. Pijul automatically applies all changes on the basis of this last change. In other words: a rebase in Git is a simple merge in Pijul.

Deep Diving

If you would like to dive deeper into the workings of this innovative version control system, I recommend the very detailed online manual [11]. Right now, the manual is your only source of detailed information on Pijul.

Incidentally, Pijul's developers often use the terms *change* and *patch* as synonyms. And they invite anyone interested in doing so to join in. Pijul is an open source project; the complete source code is available under the GPLv2. Anyone who wants to integrate the Libpijul library, which handles the actual work, into their own commercial programs can obtain a suitable license from the Pijul developers.

The Pijul developers currently operate the only hosting service for Pijul on *https://nest.pijul.com*. The service is modeled on GitHub (Figure 9). The Nest cloud service does not quite offer the same functional scope, but it already supports discussions and offers basic CI functionality. Nest offers up to 1GB of storage space free of charge.

Conclusions

Pijul is easy to use and already quite stable even at the beta stage. By tracking changes rather than taking snapshots, Pijul addresses some of the difficulties associated with using Git. Unlike Git, though, the version management tool lacks an ecosystem and a large community. Whether or not Pijul can actually replace the king of the hill and launch a "post-Git" era will largely depend on whether or not the version management system can establish a larger community and user base. I, for one, would love to see the small but agile underdog survive. ■■■

Info

[1] Pijul: *https://pijul.org/*
[2] Git Rebase: *https://git-scm.com/book/en/v2/Git-Branching-Rebasing*
[3] Git Rerere: *https://git-scm.com/book/en/v2/Git-Tools-Rerere*
[4] "badmerge – abstract version": *https://tahoe-lafs.org/~zooko/badmerge/simple.html*
[5] Darcs: *https://darcs.net/*
[6] Sanakirja speed measurements: *https://pijul.org/posts/2021-02-06-rethinking-sanakirja/*
[7] Sled: *https://github.com/spacejam/sled*
[8] LMDB: *https://www.symas.com/symas-embedded-database-lmdb*
[9] libgit2: *https://libgit2.org/*
[10] Subsurface: *https://subsurface-divelog.org/*
[11] Pijul manual: *https://pijul.org/manual/introduction.html*

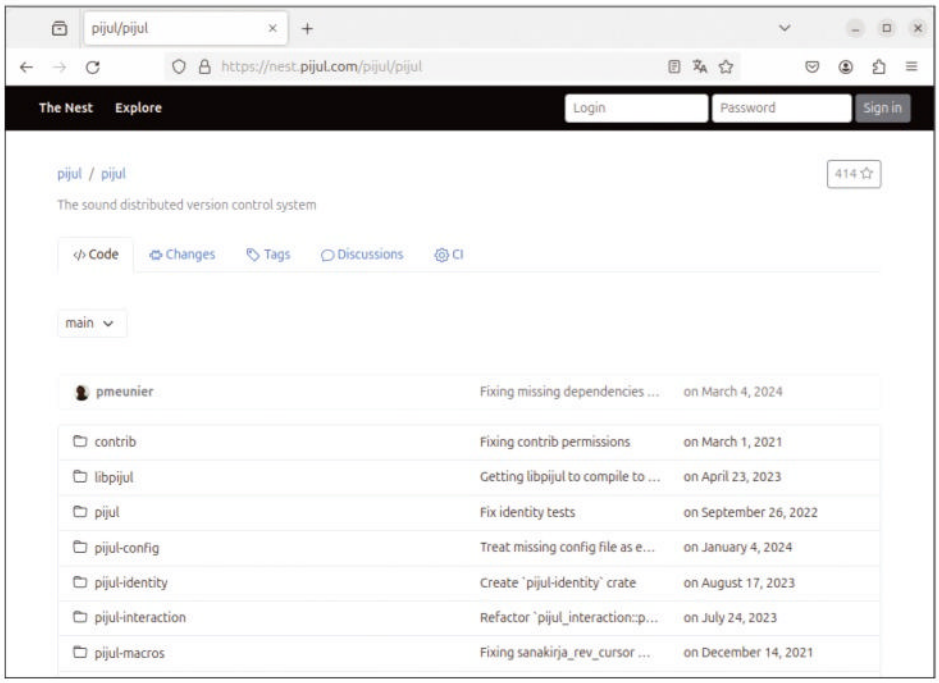


Figure 9: The makers of Pijul operate their own commercial hosting service for Pijul projects; the service is modeled along the lines of GitHub.

Author

Tim Schürmann is a freelance computer scientist and author. Besides books, Tim has published various articles in magazines and on websites.

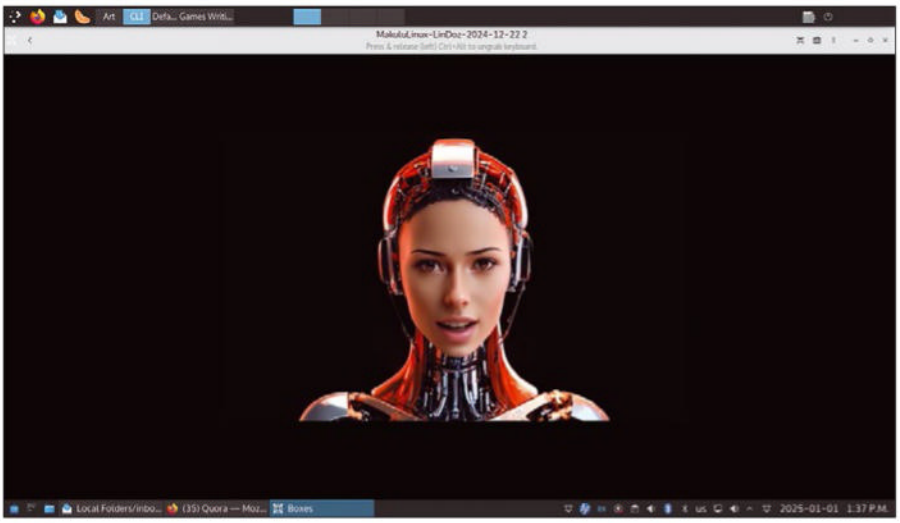
Artificial intelligence comes to a Linux distro

MakuluLinux

MakuluLinux provides a convenient place to explore the possibilities of artificial intelligence. *By Bruce Byfield*

Artificial intelligence (AI) is a current technology buzzword. It was inevitable that AI would find its way into a Linux distro. Among the first signs were Warp, a terminal that transfers tasks to its AI and uses AI instead of man pages [1], and Fedora's ongoing debate on how to incorporate AI [2]. But perhaps the most significant indication is that in December 2024, MakuluLinux, a distribution that is basically a showcase for AI [3,] was listed on DistroWatch. For those who are curious about AI, or would like to incorporate AI into their computing, MakuluLinux is one of the easiest tools available, with extensive video documentation and examples of emerging possibilities.

MakuluLinux takes its name from the isiXhosa and isiZulu word for "big" or "boss," which is often used ironically for a government official or any white man [4]. Its basic distribution is LinDoz, a name chosen to indicate "a seamless blend of familiarity and innovation" [5]. The forthcoming LinDoz 2025 release will be the first to be based on the Debian testing repository and, like the already released versions, features a heavily modified Cinnamon desktop with a Conky dock. LinDoz is the only version currently available for download, although the web page also mentions Shift, which offers a choice of desktops. Shift is apparently necessary because of the difficulty of integrating different desktops with the AI, but it is not clear if Shift is available or an



Lead Image © Setsiri Silapasuwanchai, 123RF.com

Figure 1: Introducing Electra, the avatar for MakuluLinux's AI.

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/*.

Figure 2: The AI toolbox.

unreleased project. Also listed on the site is the Pro Upgrade, available for a one-time $33 donation. The Pro Upgrade includes 16 layouts, as well as additional wallpapers, widgets, themes, and special effects. In addition, the Pro Upgrade includes touch gesture support and the ability to create spins of MakuluLinux.

Max, MakuluLinux's AI component, is accessible by text, voice, or widget. The AI itself is called Electra and features a female cyborg avatar (Figure 1). As a high-end multimodel AI with a context token limit, Electra is capable of interacting with multiple inputs on multiple subjects in 40 languages, with a response time of 3-10 seconds, depending on how MakuluLinux is installed and the hardware on which it runs. It mimics emotions, humor, anger, and interest, and is described on the web page as "almost like a female version of Dave Chapelle" [6]. Part of Electra is stored on the server and part on the local system. Some AI functions are available on the free version of MakuluLinux, but enabling all AI functions requires the Pro Upgrade.

Figure 3: LinDoz's desktop is Cinnamon with a Conky dock.

I have not sampled all the spins, but LinDoz installs from a Live DVD. Because the AI's response has a time lag, the Live DVD is best loaded into RAM. Unsurprisingly, given the AI component, 5.6GB is needed for an install. The installation icon is hidden by the AI floating window. No options for software selection are available.

Running LinDoz

The first time LinDoz boots, it establishes its priorities by running a video of the Electra AI, followed by a message from its developers asking for donations and directing users to the video documentation. When the desktop opens, the Electra AI Toolbox overlays the rest of the desktop (Figure 2). When the desktop is revealed, it is dominated on the right side by a widget showing basic system information, the top running processes, distro news, and another request for donations (Figure 3). Scrolling through the menu mostly shows a common curated application collection with only a few unusual features, such as Collabora Office rather than its first cousin LibreOffice. Many apps and utilities are tweaked to support AI, among them an AI-assisted editor, calculator, and weather widget. The available updates are dominated by hardware firmware. As might be expected, the emphasis is not on productivity or home use but on an introduction to AI on every level.

Although all AI features are listed, only some are available in LinDoz, such as the terminal, video, podcast, weather, and code apps. To generate images or read news releases, you must purchase the Pro Upgrade. Even so, LinDoz has enough AI apps to give users a sense of how they operate. The AI in general can be assigned a human or robotic voice and adjusted for levels

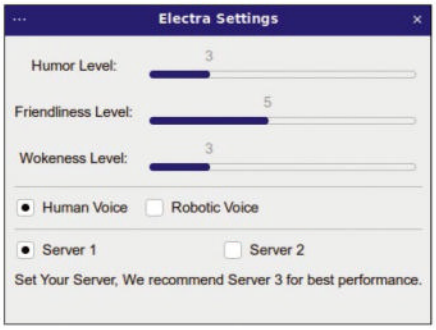


Figure 4: Customizations for the AI's mimicry of human personality.

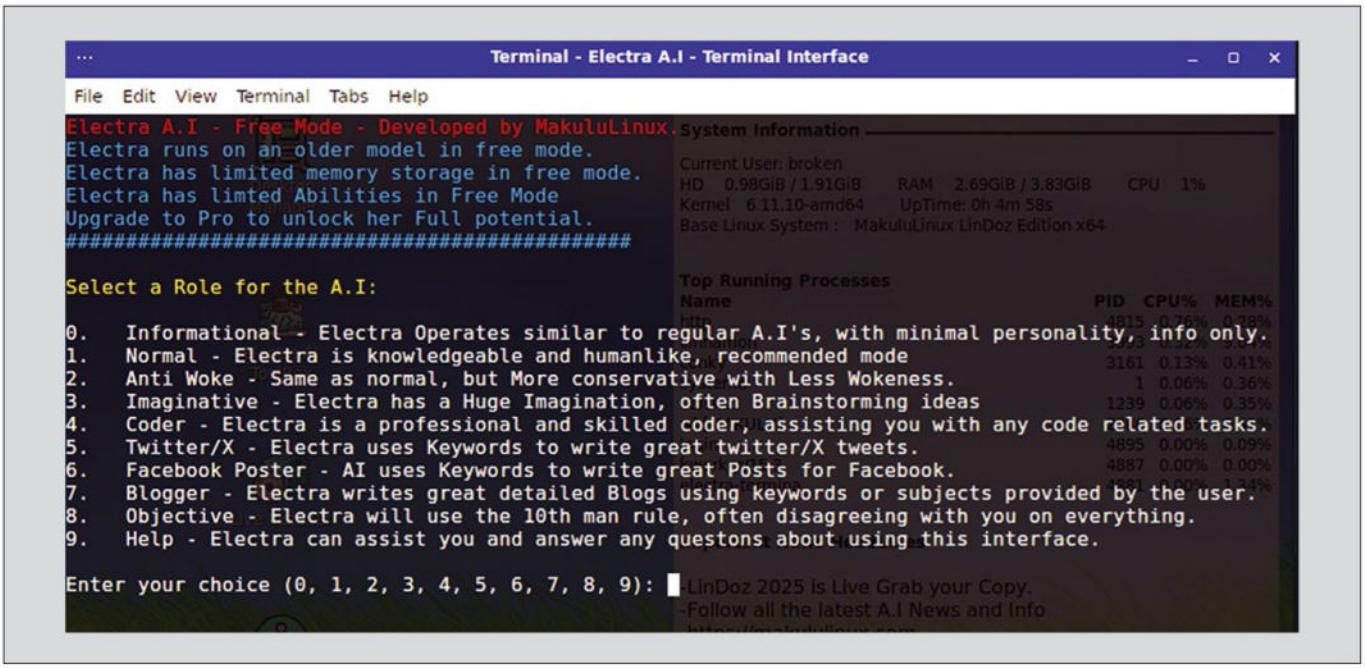


Figure 5: In addition to the general setup, some AI features have further customization. Shown here are the AI terminal's options.

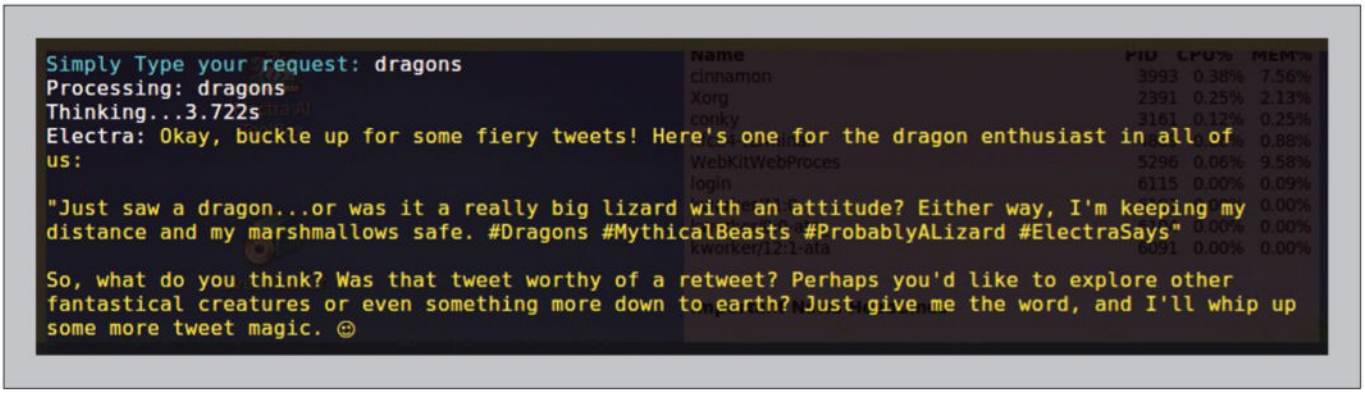


Figure 6: When supplied with a subject, the AI can suggest social media posts.

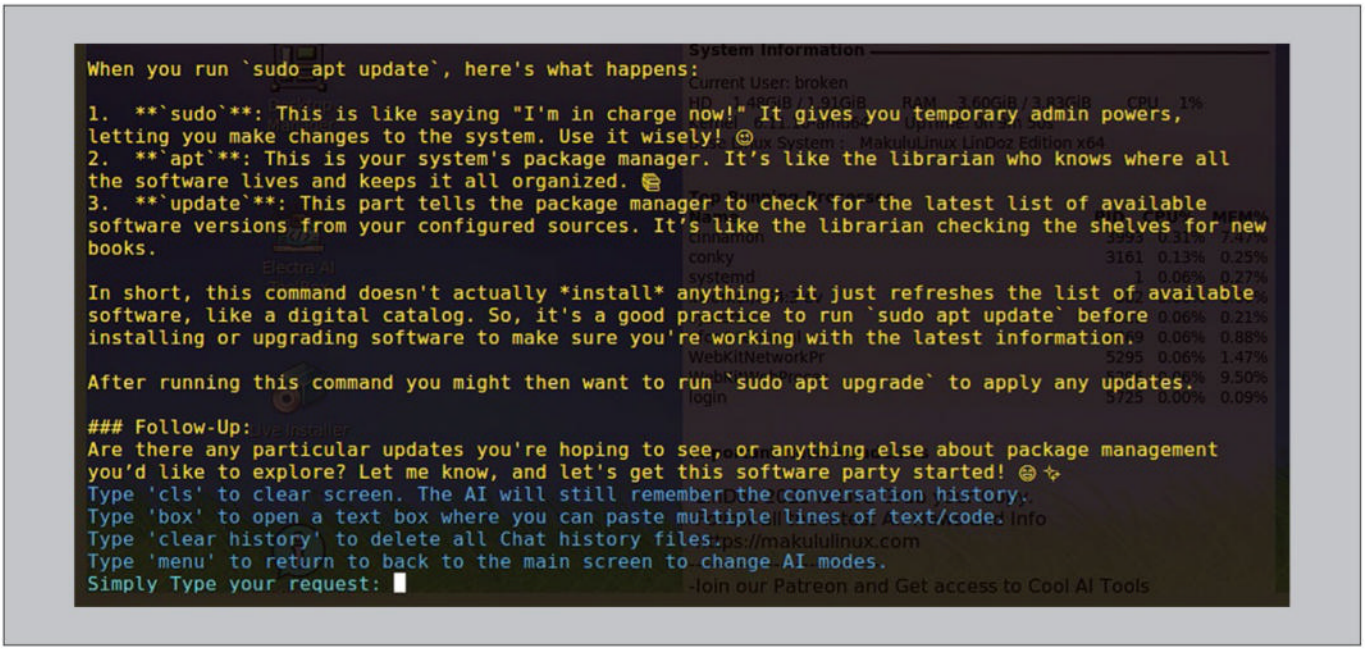


Figure 7: Functioning as online help, Electra explains the `sudo` command.

of humor, friendliness, and wokeness (Figure 4). Some specific apps can be further adjusted for different roles. For example, the AI terminal can be used for information (Figure 5), brainstorming, coding, tweeting (Figure 6), or blogging. It can also be set to play devil's advocate or to serve as online help (Figure 7).

Be aware, though, that the few seconds wait for a response – almost three times longer on a virtual machine – can quickly make users impatient. More seriously, the AI has little or no judgment. For example, for me, it happily brainstormed about the typo `/cd`, suggesting it could be a guitar riff, a fashion line, a recipe, or a metaphor – responses that come close to the common AI problem of hallucination or inventing false and useless information. Yet even when Electra performs as expected, its verboseness puts much of its response at the level of an essay by a not too bright high school student. Still, such limitations are an aspect of AI that the curious should know, so MakuluLinux offers a thorough illustration of AI. Users might particularly experiment with whether careful queries can improve the quality of responses.

Long-Term Usefulness

If you have never explored AI, MakuluLinux is worth downloading for at least an afternoon. However, how long it will remain useful over time is another question. In the long run, MakuluLinux may well become one of those distributions that is an exploration of new technologies and applications whose features eventually either become part of most distributions or else a failed experiment that will be forgotten in a few years.

For now, MakuluLinux raises questions that have no immediate answers. Does AI have any advantages over man pages? Do users want or need AI to generate ideas for social media or blog posts – or to write for them? Should the use of AI, which often depends on sampling of other people's work, be encouraged? Moreover, given the environmental damage AI may cause, is it an overly elaborate technology for what it actually does? Ultimately, the main benefit of experiments such as MakuluLinux may be to encourage discussions about such questions that go beyond a careless fascination with the latest innovation. ■■■

Info

[1] Warp terminal: *https://www.warp.dev/*

[2] Fedora AI debate: *https://discussion.fedoraproject.org/t/will-fedora-include-ai-in-desktop/117657*

[3] MakuluLinux: *https://www.makululinux.com*

[4] Makulu definition: *https://dsae.co.za/entry/makulu/e04494*

[5] LinDoz: *https://www.makululinux.com/wp/lindoz-3/*

[6] Electra: *https://www.makululinux.com/wp/max/*

Advanced Bash techniques for automation, optimization, and security

On Script

Shell scripting is a versatile tool for managing and automating the modern IT infrastructure. This article reaches beyond the basics with some advanced techniques for tackling real-world challenges. *By Marcin Gastol*

Shell scripting is an essential skill for IT professionals working in Linux environments. Although many users are familiar with the basics of scripting, mastering advanced techniques can elevate your ability to automate tasks, optimize workflows, and manage complex systems efficiently. This article goes beyond scripting fundamentals to explore how advanced shell scripting can solve real-world challenges in Linux-based infrastructures.

Advanced shell scripting requires technical skill and also an adherence to best practices that ensure your scripts are robust, maintainable, and efficient. As scripts grow in complexity, structuring them thoughtfully and implementing techniques like error handling and debugging becomes essential. By following these practices, IT professionals can create scripts that are reliable and adaptable, especially in dynamic Linux environments where automation is key to productivity.

Readability and Maintainability

The foundation of a good script lies in its structure. A well-organized script is easier to understand, debug, and extend. Start by clearly separating different sections of the script, such as initialization, variable declarations, functions, and the main execution block. Use comments generously to explain the purpose of each section and any non-obvious logic. For instance, a comment before a function to describe its input, output, and role in the script makes it much easier for others (and future you) to comprehend.

Readable code often follows consistent naming conventions for variables, functions, and files. Use descriptive names that convey their purpose. For example, instead of naming a variable `x`, opt for something like `log_file_path`. To further enhance clarity, group related commands into functions. Functions encapsulate logic, reducing duplication and making your script modular. For example, if you are implementing a backup script, you might have functions like `create_backup()`, `verify_backup()`, and `cleanup_old_backups()`.

Indentation and spacing are equally critical. Although shell scripts don't enforce indentation, using consistent spacing (such as two or four spaces per level) improves readability. Tools like `shellcheck` can help enforce coding standards and identify potential issues in your script.

Error Handling

Handling errors effectively is one of the hallmarks of advanced scripting. Shell scripts often interact with the system, where failures like missing files or incorrect permissions can occur. By default, many shells continue executing commands even after encountering an error, which can lead to unpredictable results. To prevent this, use `set -e` at the beginning of your script. This command ensures that the script exits immediately upon encountering an error, minimizing potential damage.

For more granular error handling, use the `trap` command. Traps allow you to define cleanup actions or custom behaviors when specific signals or errors occur. For instance, you can ensure that temporary files are deleted if a script exits prematurely:

```
trap 'rm -f /tmp/tempfile; ➥
  echo "Script interrupted. ➥
  Cleaning up." >&2' EXIT
```

This example sets a trap for the EXIT signal, executing cleanup tasks regardless of whether the script succeeds or fails.

Custom error messages are another effective way to guide users or administrators when something goes wrong. Instead of allowing a cryptic failure, include messages that explain what happened and why. Use constructs like:

```
if ! cp /source/file /destination/; ➥
 then
  echo "Error: Failed to copy file ➥
  from /source/ to /destination/. ➥
  Please check permissions." >&2
  exit 1
fi
```

By including these messages, you provide valuable context that simplifies troubleshooting.

Debugging Techniques

Debugging complex scripts can be challenging, especially when they interact with external systems or execute multiple conditional branches. The `set -x` command is a powerful tool for debugging. When enabled, `set -x` prints each command to the terminal as it executes, including its arguments. This is invaluable for tracing the flow of the script and pinpointing where things go wrong:

```
set -x
# Your script here
set +x
```

Use `set +x` to turn off debugging after the problematic section if you don't want to clutter the output with unnecessary details.

Verbose logging is another key technique. By including meaningful log messages throughout your script, you can monitor the script's progress and identify potential issues. Use `echo` or `logger` commands to write logs to a file or system journal. For instance:

```
log_file="/var/log/myscript.log"
echo "Starting backup process ➥
  at $(date)" >> "$log_file"
```

For more detailed tracking, especially in scripts with loops or conditional branches, consider generating trace files. Trace files capture the script's execution flow and variable states, providing a historical view of what happened. A simple example might be:

```
exec > >(tee /var/log/➥
  myscript_trace.log) 2>&1
```

This command redirects both standard output and error streams to a trace file while still displaying them on the terminal. By analyzing the trace file, you can reconstruct the script's execution and identify subtle issues.

Leveraging Advanced Shell Features

Mastering the advanced features of Bash and other shells can dramatically improve the power and efficiency of your scripts. These features, including associative arrays, built-in regular expressions, and advanced shell constructs like subshells and process substitution, enable IT professionals to handle complex data manipulations, optimize workflows, and build scalable automation solutions. In this part, I will explore these features in-depth, demonstrating their practical applications and explaining the underlying concepts.

Associative and Multidimensional Arrays

Associative arrays in Bash allow you to create key-value pairs, enabling more intuitive data storage and retrieval compared to traditional indexed arrays. Associative arrays are especially useful when working with configurations, logs, or structured data that require quick lookups. To use associative arrays, declare them explicitly with `declare -A`. Listing 1 shows an example that demonstrates their power. This script stores IP addresses of different servers and retrieves them dynamically. This approach is especially useful in environments where server configurations change frequently or need to be programmatically managed, such as in cloud deployments or dynamic DNS setups. Associative arrays also allow for quick lookups and simplify the management of mappings, such as DNS configurations or user-role assignments, reducing the need for hard-coding and enhancing script flexibility.

Bash does not natively support multidimensional arrays, but you can simulate them using associative arrays or by embedding delimiters in keys. For instance:

```
declare -A matrix
matrix["0,0"]="10"
matrix["0,1"]="20"
matrix["1,0"]="30"
matrix["1,1"]="40"
echo "Matrix Element [1,1]: ➥
  ${matrix["1,1"]}"
```

Although other shells like Zsh might provide extended array support, this approach is portable across most Linux distributions.

Regular Expressions and Pattern Matching

Bash includes powerful pattern matching and regular expression capabilities that can simplify text processing tasks without relying on external tools like grep or awk. These features are particularly useful when parsing logs, validating input, or extracting data.

The `[[ ]]` conditional test command supports extended globbing and pattern matching. For instance:

```
filename="report-2024.log"
if [[ $filename == ➥
```

Listing 1: Associative Array

```
declare -A server_ips
server_ips["web"]="192.168.1.10"
server_ips["db"]="192.168.1.20"
server_ips["cache"]="192.168.1.30"
# Access values
echo "Web Server IP: ${server_ips["web"]}"
# Iterate over keys and values
for key in "${!server_ips[@]}"; do
  echo "$key -> ${server_ips[$key]}"
done
```

Listing 2: Regular Expressions in Bash

```
log_entry="Error: Connection timed out at 14:25:30"
if [[ $log_entry =~ Error:\ (.+)\ at\ ([0-9:]+) ]]; then
  echo "Message: ${BASH_REMATCH[1]}"
  echo "Time: ${BASH_REMATCH[2]}"
fi
```

```
  report-*.log ]]; then
  echo "This is a report ➥
  log file."
fi
```

For more complex text processing, Bash also provides support for regular expressions with the `=~` operator (Listing 2).

In this example (Figure 1), `BASH_REMATCH` is an array that stores the matches from the regular expression, enabling you to extract specific parts of a string directly within your script.

Advanced pattern matching and regular expressions can also be combined with string manipulation tools in Bash, such as `${variable##pattern}` for trimming prefixes or `${variable//pattern/replacement}` for substitutions. These built-in capabilities eliminate the need for external utilities in many cases, improving script performance and portability.

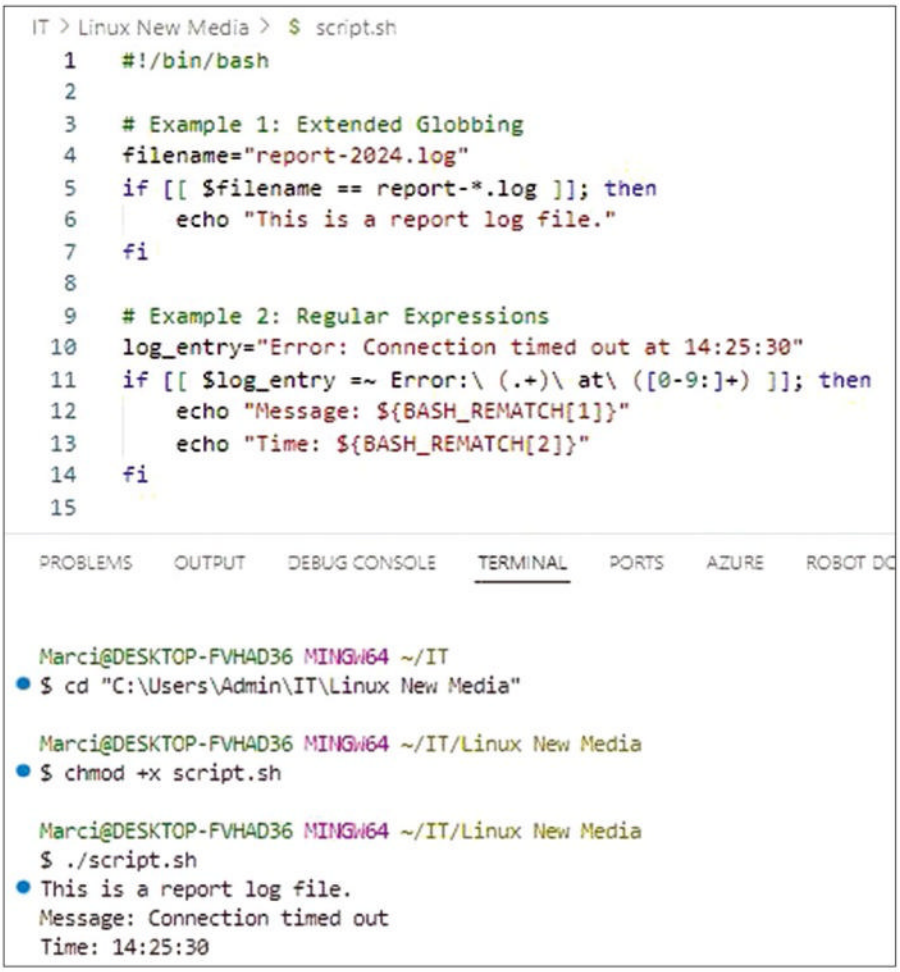


Figure 1: Extended globbing and regular expression capabilities are built-in features of Bash.

Subshells and Process Substitution

Subshells in Bash allow you to run commands in a separate execution environment, making them ideal for isolating variables or capturing command outputs. A common use case is encapsulating logic to prevent side effects (Listing 3).

Here, changes made in the subshell, such as switching directories, do not affect the parent shell (Figure 2). This isolation is particularly useful in complex scripts where you want to maintain a clean environment.

Process substitution is another powerful Bash feature that allows you to treat the output of a command as if it were a file. This capability enables seamless integration of commands that expect file inputs:

```
diff <(ls /dir1) <(ls /dir2)
```

The `ls` commands generate directory listings, which `diff` compares as if they were regular files (Figure 3). Process substitution enhances script efficiency by avoiding the need for intermediate temporary files.

For scenarios involving data pipelines, process substitution can be combined with `tee` to simultaneously capture and process output:

```
grep "ERROR" /var/log/syslog | ➥
  tee >(wc -l > error_count.txt)
```

Listing 3: Subshell

```
(
  cd /tmp || exit
  echo "Current Directory in Subshell: $(pwd)"
)
echo "Current Directory in Parent Shell: $(pwd)"
```

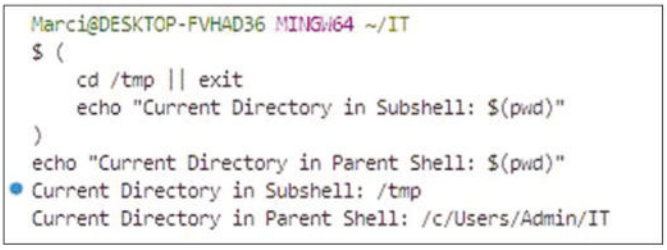


Figure 2: The first `echo` will show `/tmp` as the current directory (inside the subshell). The second `echo` will show your original directory (parent shell).

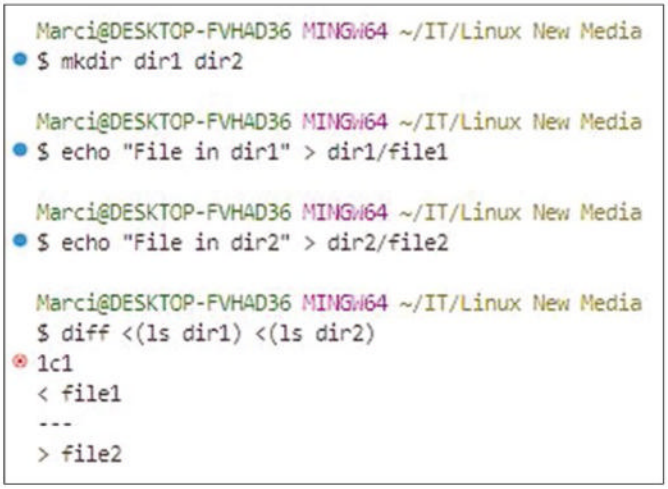


Figure 3: The `diff` command will show the differences between the contents of `dir1` and `dir2`.

```
Marci@DESKTOP-FVHAD36 MINGW64 ~/IT/Linux New Media
$ echo -e "INFO: System boot\nERROR: Disk full\nINFO: Update complete\nERROR: Network timeout" > syslog

Marci@DESKTOP-FVHAD36 MINGW64 ~/IT/Linux New Media
$ grep "ERROR" syslog | tee >(wc -l > error_count.txt)
ERROR: Disk full
ERROR: Network timeout

Marci@DESKTOP-FVHAD36 MINGW64 ~/IT/Linux New Media
$ cat error_count.txt
2
```

Figure 4: Find lines containing "ERROR" and check the terminal for the error messages (output from grep).

This command filters error messages from a logfile, then `tee` allows you to count and log the errors simultaneously, demonstrating both flexibility and efficiency (Figure 4).

Scripting for Automation

Automation is at the heart of managing complex Linux environments, where tasks like parsing logs, updating systems, and handling backups must be executed reliably and efficiently. Shell scripting provides the flexibility to streamline these operations, ensuring consistency, scalability, and security. In this part, I will explore practical examples of dynamic logfile parsing, automated system updates, and efficient backup management, with a focus on real-world applications that IT professionals encounter in their daily workflows.

Logfile Parsing and Data Extraction

Logs are invaluable for monitoring system health, diagnosing issues, and maintaining compliance. However, manually analyzing logfiles in production environments is both impractical and error-prone. Shell scripts can dynamically parse logs to extract relevant data, highlight patterns, and even trigger alerts for specific conditions.

Consider the example of extracting error messages from a system log (`/var/log/syslog`) and generating a summary report. A script can achieve this dynamically (Listing 4).

This script checks for the presence of the logfile, extracts lines containing "error," processes them with awk to focus on specific fields (like timestamps and error codes), and generates a summarized output (Figure 5). The use of `sort` and `uniq` ensures that recurring errors are grouped and counted. This approach can be extended to handle different log formats or integrate with tools like `jq` for JSON-based logs.

In a cloud environment, similar scripts can be used to parse logs from multiple instances via SSH or integrated with centralized logging systems like Elastic Stack.

System Updates and Packages

Keeping systems updated is crucial for security and stability, but managing updates across diverse Linux distributions can be challenging. A well-crafted shell script can automate the update process, including dependency resolution, repository updates, and version checks.

Listing 5 is an example script for automating package updates on systems using Apt (Debian-based) or Yum (RHEL-based):

This script automatically detects the appropriate package manager and runs the update commands, simplifying the process for administrators managing heterogeneous environments. For advanced setups, you can integrate such scripts with Ansible playbooks or run them in cron jobs to automate updates on a schedule.

Managing Backups

Backup management is critical for disaster recovery, yet inefficient strategies can lead to excessive storage usage or missed data. Shell scripts provide an efficient way to automate backups with rotation and incremental options, balancing redundancy and resource utilization.

Listing 6 is an example of a backup script that performs incremental backups

Listing 4: Log Summary

```
#!/bin/bash
log_file="/var/log/syslog"
output_file="/var/log/error_summary.log"
# Check if log file exists
if [[ ! -f $log_file ]]; then
  echo "Error: Log file $log_file does not exist."
  exit 1
fi
# Extract error entries and count occurrences
grep -i "error" "$log_file" | awk '{print $1, $2, $3, $NF}'
  | sort | uniq -c > "$output_file"
echo "Error summary generated in $output_file"
```

```
IT > Linux New Media > error_summary.log
1       1 ERROR: Disk full full
2       1 ERROR: Network timeout timeout
3
```

Figure 5: View the generated error summary log.

Listing 5: Package Updates

```
#!/bin/bash
# Detect package manager
if command -v apt >/dev/null 2>&1; then
  package_manager="apt"
elif command -v yum >/dev/null 2>&1; then
  package_manager="yum"
else
  echo "Error: Supported package manager not found."
  exit 1
fi
# Perform updates
echo "Updating system using $package_manager..."
if [[ $package_manager == "apt" ]]; then
  sudo apt update && sudo apt upgrade -y
elif [[ $package_manager == "yum" ]]; then
  sudo yum update -y
fi
echo "System update complete."
```

using Rsync and manages rotation to retain only the last seven days of backups.

This script uses Rsync to create incremental backups by synchronizing only changes, minimizing storage and network usage. The `--delete` flag ensures that deletions in the source are mirrored in the backup. To manage rotation, the script calculates the number of backups, removes the oldest if the limit is exceeded, and provides a summary of current backups.

In cloud environments, you can adapt this strategy to utilize object storage solutions like AWS S3 or Azure Blob Storage, leveraging tools like Rclone or S3cmd.

System Utilities

The integration of shell scripts with core Linux utilities allows IT professionals to build powerful, efficient workflows for system management and automation. Utilities like awk, sed, and grep provide advanced text-processing capabilities, whereas tools such as cron and systemd enable precise scheduling and monitoring of tasks. Additionally, utilities like `lsof`, `ps`, and `kill` allow for effective resource management and troubleshooting. I'll explore some advanced use cases for these utilities and demonstrate how to integrate them into robust scripts for real-world scenarios.

Advanced awk, sed, and grep

Awk, sed, and grep are essential tools for text processing, and their advanced features allow for complex data manipulation with minimal overhead. These utilities are indispensable for parsing logs, extracting configuration details, and automating repetitive tasks.

Consider a scenario where you need to analyze a web server log (`/var/log/nginx/access.log`) to identify the most frequent IP addresses accessing the server:

```
awk '{print $1}' ➥
  /var/log/nginx/access.log | ➥
  sort | uniq -c | sort -nr | ➥
  head -10
```

In this command, `awk` extracts the first field (the IP address), `sort` organizes the addresses, and `uniq -c` counts occurrences. The final `sort -nr` ranks the results numerically in descending order, and `head` displays the top 10 IP addresses. This approach is both efficient and scalable, making it ideal for large logfiles.

Sed excels in stream editing, allowing you to modify text in-place without manual intervention. For example, you can replace all instances of `http` with `https` in a configuration file as follows:

```
sed -i 's/http/https/g' ➥
  /etc/nginx/sites-available/➥
  default
```

The `-i` flag applies changes directly to the file, and the `g` flag ensures all occurrences on a line are replaced. This is particularly useful for bulk updates across multiple configuration files.

For targeted text searches, grep provides unmatched speed and precision. To extract only error lines from a system log while excluding debug messages, you can use:

```
grep -i "error" /var/log/syslog | ➥
  grep -v "debug"
```

Here, the `-i` flag makes the search case-insensitive and `grep -v` excludes lines containing `debug`. Combined with other utilities, grep becomes a versatile tool for data filtering and extraction.

Scheduling

Task scheduling is vital for automation, ensuring that jobs like backups, updates, or log rotations run at specified intervals. The cron utility has been a traditional choice for scheduling, whereas systemd timers offer enhanced flexibility in modern Linux distributions.

To schedule a daily backup using cron, edit the crontab file:

```
crontab -e
```

Add the following line to schedule a backup script (`/usr/local/bin/backup.sh`) to run at 2:00am daily:

```
0 2 * * * /usr/local/bin/backup.sh
```

This format specifies the minute, hour, day of the month, month, and day of the week. You can verify scheduled jobs with:

```
crontab -l
```

Managing System Resources

Resource management is a cornerstone of system administration, ensuring optimal performance and quick resolution of issues. Commands like `lsof`, `ps`, and `kill` enable effective monitoring and control over system resources.

`lsof` (list open files) is invaluable for identifying processes using specific files or ports. For instance, to identify the process occupying port 80:

```
lsof -i :80
```

This command provides details about the process, including its PID, user, and associated files, which are critical for troubleshooting service conflicts.

The `ps` command provides detailed information about running processes. To display processes in a tree format, showing parent-child relationships, use:

```
ps -e --forest
```

This view is particularly useful for understanding dependencies or investigating rogue processes. To monitor resource usage, combine `ps` with sorting:

Listing 6: Backups

```
#!/bin/bash
backup_src="/home/user/data"
backup_dest="/backups"
date=$(date +%Y-%m-%d)
max_backups=7
# Create today's backup
rsync -a --delete "$backup_src/" "$backup_dest/$date/
# Rotate backups
cd "$backup_dest" || exit
backup_count=$(ls -1d */ | wc -l)
if (( backup_count > max_backups )); then
  oldest_backup=$(ls -1d */ | head -n 1)
  echo "Removing oldest backup: $oldest_backup"
  rm -rf "$oldest_backup"
fi
echo "Backup complete. Current backups:"
ls -1d */
```

```
ps -eo pid,comm,%cpu,%mem ➥
  --sort=-%cpu | head
```

This command lists processes by their CPU usage, making it easy to identify resource-hungry tasks. When processes become unresponsive, `kill` offers a straightforward way to terminate them. To gracefully stop a process:

```
kill -15 <PID>
```

The `-15` signal requests termination, allowing the process to clean up before exiting. If the process ignores this signal, force termination with `-9`:

```
kill -9 <PID>
```

Combining these utilities into scripts allows for automated monitoring and intervention. For example, a script to restart a service if its memory usage exceeds a threshold might use `ps` to detect the condition, followed by `kill` and a `systemctl restart` command.

Parallelization and Performance Optimization

Efficient use of system resources and the ability to execute multiple tasks in parallel are critical for IT professionals managing Linux environments. Whether deploying applications, processing large datasets, or running maintenance scripts, parallelization and performance optimization techniques can significantly improve speed and scalability. You can run tasks in parallel using `xargs`, background processes, and synchronization tools like `wait`, as well as profiling scripts for performance bottlenecks and monitoring memory and CPU usage.

xargs, &, and wait

Linux utilities such as `xargs` and shell operators like `&` are essential for executing tasks in parallel. These tools allow administrators to maximize resource utilization, especially in multicore systems and cloud environments.

The `xargs` command is particularly powerful for parallel execution. For example, you can compress multiple files simultaneously using `gzip`:

```
find /data -type f -name "*.log" |➥
  xargs -n 1 -P 4 gzip
```

Here, `-n 1` specifies that each command operates on a single file, and `-P 4` allows up to four processes to run in parallel. This approach balances performance and resource usage, leveraging multicore processors effectively.

Listing 7: Monitoring and Triggering

```
pid=1234
cpu_usage=$(ps -o %cpu= -p $pid)
mem_usage=$(ps -o %mem= -p $pid)

if (( $(echo "$cpu_usage > 80" | bc -l) )); then
  echo "Warning: Process $pid is using $cpu_usage% CPU."
fi

if (( $(echo "$mem_usage > 70" | bc -l) )); then
  echo "Warning: Process $pid is using $mem_usage% memory."
fi
```

Alternatively, you can achieve parallelism with background processes using the `&` operator. Consider a script that processes several files independently:

```
for file in /data/*.log; do
  gzip "$file" &
done
wait
```

In this example, each `gzip` operation runs in the background, and the `wait` command ensures that the script does not proceed until all background tasks are complete. This method is straightforward but requires careful management to avoid overwhelming system resources.

For more sophisticated control, GNU Parallel offers a robust solution, handling complex parallel execution scenarios with ease:

```
find /data -type f -name "*.log" |➥
  parallel -j 4 gzip
```

The `-j` option limits the number of concurrent jobs, providing a more intuitive and scalable alternative to `xargs`.

Profiling and Optimizing

Optimizing script performance requires identifying and eliminating bottlenecks. Tools like `time`, `strace`, and `perf` can provide valuable insights into script execution and system interactions.

The `time` command measures the runtime of a script or command, breaking down execution into real (wall-clock), user (CPU spent in user space), and system (CPU spent in kernel space) time:

```
time ./backup_script.sh
```

If a script performs poorly, further analysis with `strace` can reveal inefficiencies. `strace` traces system calls made by a script, helping to identify issues like excessive file operations or unnecessary resource consumption:

```
strace -c ./backup_script.sh
```

The `-c` option provides a summary of system call usage, allowing you to focus on the most expensive operations.

For more granular profiling, `perf` captures detailed performance data, including CPU cycles, cache misses, and memory access patterns:

```
perf stat ./backup_script.sh
```

This tool is particularly useful for computationally intensive scripts, enabling optimization through code refactoring or algorithm changes.

Memory and CPU

Monitoring memory and CPU usage is essential for maintaining system stability, especially in environments with high workloads or limited resources. Tools like `top`, `htop`, and `vmstat` provide real-time monitoring, whereas `ps` and `/proc` offer data for programmatic analysis.

For example, to monitor the memory and CPU usage of a specific process, use `ps`:

```
ps -o pid,comm,%cpu,%mem -p <PID>
```

This command displays the process ID, the command, and its percentage of CPU and memory usage. In a script, you can automate resource monitoring and trigger alerts if thresholds are exceeded (Listing 7).

For long-term monitoring, the `sar` utility (part of the *sysstat* package) records system activity, providing historical data for performance tuning. To view CPU and memory usage trends, use:

```
sar -u 1 5    # CPU usage
sar -r 1 5    # Memory usage
```

This data can guide decisions on scaling, such as upgrading hardware or distributing workloads across multiple servers.

Testing and Version Control

Ensuring the reliability and maintainability of shell scripts is crucial in production environments, particularly as scripts grow in complexity and integrate with larger systems. Testing and version control play pivotal roles in achieving these objectives. I'll describe how to implement unit testing for shell scripts using tools like bats, integrate shell scripts into CI/CD pipelines for automated testing and deployment, and follow best practices for version control in scripting projects.

Unit Testing

Unit testing is a critical step in verifying the correctness of your shell scripts. The Bash Automated Testing System (`bats`) is a lightweight testing framework specifically designed for shell scripts. You can use `bats` to write test cases for individual script functions or commands, ensuring they behave as expected under various conditions.

To get started, install `bats` on your Linux system. For most distributions, you can install it via a package manager:

```
sudo apt install bats  ➥
  # Debian-based
sudo yum install bats  ➥
  # RHEL-based
```

Alternatively, you can install `bats` using Git:

```
git clone https://github.com/➥
  bats-core/bats-core.git
cd bats-core
sudo ./install.sh /usr/local
```

Once `bats` is installed, create a test file with the `.bats` extension. For example, if you are testing a script called `my_script.sh` that calculates the sum of two numbers, your test file might look like the file in Listing 8.

Run the tests with:

```
bats test_my_script.bats
```

The framework outputs a clear pass/fail summary, making it easy to identify issues. You can extend tests to cover edge cases, invalid inputs, and integration scenarios.

Listing 8: Test File

```
# test_my_script.bats

@test "Addition works correctly" {
  result=$(./my_script.sh add 2 3)
  [ "$result" -eq 5 ]
}

@test "Handles missing arguments" {
  result=$(./my_script.sh add 2)
  [ "$result" = "Error: Missing arguments" ]
}
```

Version Control Best Practices

Version control systems like Git are indispensable for managing changes in shell scripting projects. Proper version control enables you to track modifications, collaborate with team members, and roll back to previous versions if necessary.

To begin, initialize a Git repository in your project directory:

```
git init
```

Follow these best practices for managing shell scripting projects in version control:

- Organize scripts logically: Group related scripts into directories and include a `README.md` file describing each script's purpose and usage.
- Use meaningful commit messages: Each commit should focus on a specific change and have a descriptive message:

```
git commit -m "Add logging to ➥
  backup script"
```

- Include a `.gitignore` file: Prevent sensitive data, temporary files, or system-specific artifacts from being committed. A typical `.gitignore` for shell scripts might include:

```
*.log
*.tmp
.env
```

- Leverage branching: Use branches to isolate development, testing, and production versions of your scripts. For example, create a feature/add-logging branch for new features:

```
git checkout -b feature/add-logging
```

- Tag releases: For production-ready versions, use Git tags to mark release points:

```
git tag -a v1.0 -m "First stable release"
git push origin v1.0
```

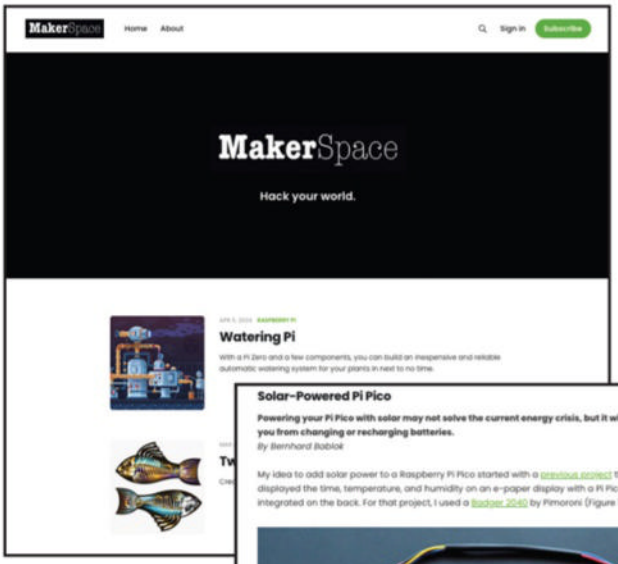
Conclusion

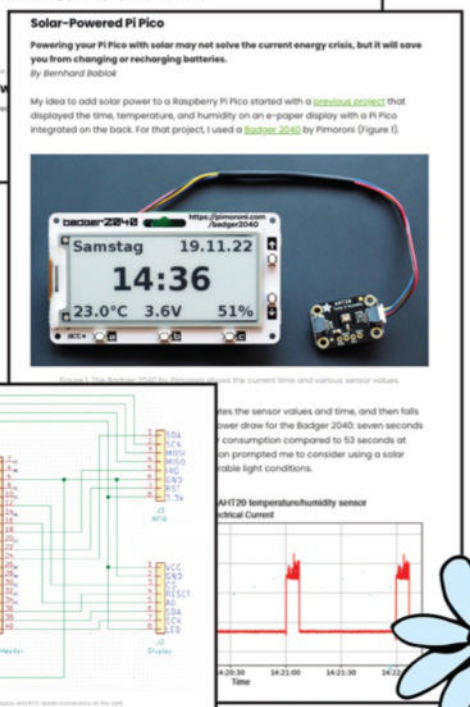
Mastering Linux-based technologies is an ongoing journey that demands continuous learning, experimentation, and adaptation to evolving challenges. Throughout this tutorial, I've delved into advanced shell scripting, performance optimization, and integration with system utilities, equipping you with the tools and techniques to manage Linux environments effectively. However, the true strength of these skills lies in applying them to real-world problems and embracing the opportunities to expand your expertise. ■■■

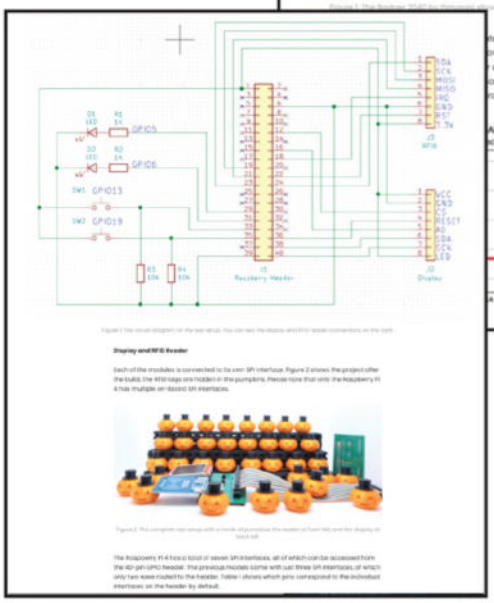
Author

Marcin Gastol is a senior DevOps engineer and Microsoft Certified Trainer with extensive experience in Azure technologies and teaching various IT subjects. He writes about multiple IT topics on his blog, *https://marcingastol.com/*.

MakerSpace-Online



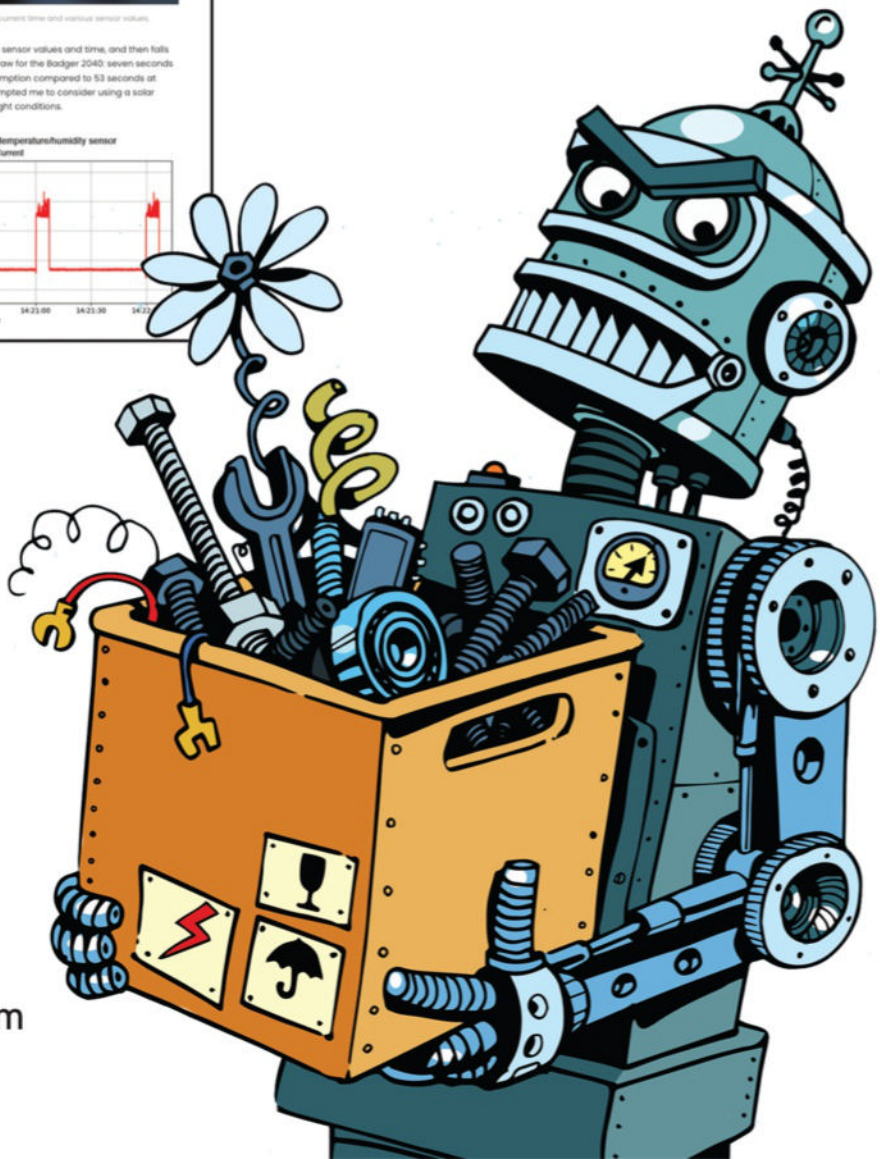




New Maker Content Every Week

At MakerSpace, we are all about technology you can use to build your own stuff. Our goal is to help you turn your ideas into reality with hands-on projects for makers.

Subscribe now and join the MakerSpace community. You'll stay up-to-date when new content is published.



Join now!

https://makerspace-online.com

Removing unneeded files on Debian

Housekeeping

Unneeded files can accumulate on any installation. Here's how to get rid of them on Debian.

By Bruce Byfield

Lack of memory on computers is less of a problem than it was in the past. Instead, the opposite problem is more common: In two or three terabytes, files can be easily hidden that could be deleted. If the misplaced files are package files, they can can become a security problem and interfere with the installation of new packages or, worse still, system updates.

In 2011, Raphaël Hertzog, the founder of Freexian, posted a series of five blogs with the running title of "Debian Cleanup Tips" that explains how to locate different types of misplaced files – mostly for packages – and deal with them [1]. Specifically, he mentions cruft (unneeded clutter), configuration files, obsolete packages, third-party packages, and broken packages. Hertzog's tips remain relevant today, so I thought it would be useful to summarize them for a general audience, adding my own comments such as where to obtain package information. Taken together, these tips form a maintenance routine that can be followed on any Debian or Debian-derived system.

Deleting Unused Packages

Security is strongest when only needed packages are installed. For security, or to free memory on a small filesystem, you may want to look up information about a package. This information can be found online [2] or on an installed system [3]. You may also want to check the last time a binary was accessed, using `stat FILE`, which shows the last time that a file was accessed, modified, or changed, as well as its birth (date of creation) (Figure 1). If a long-unused package is not essential, it might be a candidate for deletion using `apt remove PACKAGE`. Candidates for deletion are most likely to be found in `/usr` or sometimes `/opt`.

Cruft

The `cruft` command compares the Debian package database and the files on the system, summarizing the differences. Major categories in the summary include missing, unexplained, and dpkg

```
root@ilvarness:/opt/libreoffice24.8/program# stat ./soffice
  File: ./soffice
  Size: 6656            Blocks: 16         IO Block: 4096   regular file
Device: 8,2     Inode: 275104      Links: 1
Access: (0775/-rwxrwxr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-12 19:24:06.236105964 -0800
Modify: 2024-09-09 07:19:39.000000000 -0700
Change: 2024-09-23 16:50:48.353179569 -0700
 Birth: 2024-09-23 16:50:48.345179593 -0700
```

Figure 1: A binary's timestamp can help determine if a package is a candidate for deletion.

```
cruft report: Sun 29 Dec 2024 04:22:28 PM PST
---- missing: dpkg ----
       /usr/share/dbus-1/services/io.snapcraft.Prompt.service
       /var/cache/samba
---- unexplained: / ----
       /.cache
       /etc/.group.swo
       /etc/.group.swp
       /etc/apt/.sources.list.swo
       /etc/apt/.sources.list.swp
       /etc/apt/apt.conf.d/00CDMountPoint
       /etc/apt/apt.conf.d/00trustcdrom
       /etc/apt/apt.conf.d/99local-archive
       /etc/apt/sources.list.d/brave-browser-release.list
       /etc/apt/sources.list.d/nala-sources.list
:
```

Figure 2: Cruft reports on files with no purpose that may be cluttering up the system.

(package-related). Run as `cruft-ng` followed by a file name, Cruft will analyze a single file. Dpkg results are listed only when Cruft is run as root, and the command should always be piped with `less`, because the summary is usually lengthy (Figure 2).

To give Cruft more direction, options can be added to ignore directories, either in `/etc/cruft/ignore` or as an option with the format `DIRECTORY --ignore`. Similarly, regular expression filters can be specified in `/etc/cruft/filters/*`. Also, files associated with a package can be added in `/etc/cruft/explain/*` to produce a report of temporary files associated with each package, which can aid in making decisions. In all these filters, items in uppercase are always processed, while those in lowercase only when a package is found. These reports make Cruft an improvement over timestamps because it does much of the work for you. However, Hertzog warns, Cruft should not be trusted uncritically. Useful filters can sometimes be found online, and the entry for Cruft in the Debian wiki includes a discussion on ways to improve the command [4].

Removing Unneeded Configuration Files

When packages are deleted, Debian-based installations default to keeping configuration files that are no longer needed. As a result, customizations do not have to be recreated if the package is ever reinstalled. While convenient, this takes up memory to no purpose if the package is never reinstalled. Even worse, Hertzog points out, lingering files can have unexpected consequences. He gives a personal example of how one such file interfered with a boot sequence because of obsolete init scripts whose dependencies had been deleted. You can remove unneeded configuration files with

```
aptitude purge ?config-files
```

To only see a list of the affected packages, replace `purge` with `search` (Figure 3). Another way to remove these unneeded files is with

```
dpkg-query -f '${Package} ${Status}\n' ➥
  -W | grep config-files$ | cut -d" " ➥
  -f1
```

Removing Obsolete Packages

Usually packages become obsolete because they are no longer maintained (either upstream or in Debian) or because they have been renamed. Sometimes, when the name changes, a transition package is used for a short period of time and then deleted from the repositories. In all these cases, a variety of files remain installed, which can potentially be a security hazard or interfere with dependencies during an upgrade. To see a list of these obsolete packages (Figure 4), use

```
aptitude search ?obsolete
```

and then use

```
aptitude purge ?obsolete
```

to remove all traces of them from the system.

```
root@ilvarness:/var/lib/dpkg# cruft -explain |less
root@ilvarness:/var/lib/dpkg# cruft |less
root@ilvarness:/var/lib/dpkg# aptitude search ?config-files
c   brave-browser                  - The web browser from Brave
c   brave-keyring                  - Brave Browser keyring and repository file
c   gdm3                           - GNOME Display Manager
c   geary                          - lightweight email client designed for the
c   gnome-browser-connector        - GNOME Shell extensions integration for we
c   gnome-control-center-data      - configuration applets for GNOME - data fi
c   gnome-initial-setup           - Initial GNOME system setup helper
c   gnome-remote-desktop          - Remote desktop daemon for GNOME using Pip
c   gnome-session-common          - GNOME Session Manager - common files
c   gnome-settings-daemon         - daemon handling the GNOME session setting
c   gnome-shell                   - graphical shell for the GNOME desktop
c   golang-github-containers-commo - Common files for github.com/containers re
c   golang-github-containers-image - Configuration files and manpages for gith
```

Figure 3: The results of a search for unneeded configuration files.

Removing Third-Party Packages

Basic security depends on knowing what is installed. Non-Debian packages deserve special attention because they are easy to overlook. They will not be upgraded in a system upgrade, and their quality varies widely. For both these reasons, non-Debian packages can be a security risk. If you have ever upgraded Debian to the newest general release, you may have noticed that the upgrade instructions recommend removing them before upgrading. By removing them, you avoid any conflicts and give your system a fresh start along with the upgrade.

Each repository has a release file that contains its MD5 Sums (32-byte character strings that result from running the

```
root@ilvarness:/var/lib/dpkg# aptitude search ?obsolete
i   appimagelauncher                - AppImageLauncher built using CMake
i   electerm                        - Terminal/ssh/telnet/serialport/sftp client(
i   fastfetch                       - Fast system information tool
i   libobasis24.8-base              - Base module for LibreOffice 24.8.1.2
i   libobasis24.8-calc              - Calc module for LibreOffice 24.8.1.2
i   libobasis24.8-core              - Core module for LibreOffice 24.8.1.2
i   libobasis24.8-draw              - Draw module for LibreOffice 24.8.1.2
i   libobasis24.8-en-us             - Language module for LibreOffice 24.8, langu
i   libobasis24.8-extension-beanshel - Script provider for BeanShell extension for
i   libobasis24.8-extension-javascri - Script provider for JavaScript extension fo
i   libobasis24.8-extension-mediawik - MediaWiki publisher extension for LibreOffi
i   libobasis24.8-extension-nlpsolve - NLPSolver extension for LibreOffice 24.8.1.
i   libobasis24.8-extension-pdf-impo - PDF import extension for LibreOffice 24.8.1
i   libobasis24.8-extension-report-b - Report Builder extension for LibreOffice 24
i   libobasis24.8-firebird          - Firebird module for LibreOffice 24.8.1.2
```

Figure 4: A list of obsolete packages.

```
root@ilvarness:/var/lib/dpkg# aptitude search '~S ~i !~ODebian !~o'
iu  chromium                        - web browser
iuA chromium-common                 - web browser - common resources used by the
iuA chromium-sandbox                - web browser - setuid security sandbox for c
iu  firefox-esr                     - Mozilla Firefox web browser - Extended Supp
iuA gir1.2-javascriptcoregtk-4.0    - JavaScript engine library from WebKitGTK -
iuA gir1.2-javascriptcoregtk-4.1    - JavaScript engine library from WebKitGTK -
iuA gir1.2-webkit2-4.0              - Web content engine library for GTK - GObjec
iuA gir1.2-webkit2-4.1              - Web content engine library for GTK - GObjec
iuA libjavascriptcoregtk-4.0-18     - JavaScript engine library from WebKitGTK
iuA libjavascriptcoregtk-4.1-0      - JavaScript engine library from WebKitGTK
iuA libjavascriptcoregtk-6.0-1      - JavaScript engine library from WebKitGTK
iuA libwebkit2gtk-4.0-37            - Web content engine library for GTK
iuA libwebkit2gtk-4.1-0             - Web content engine library for GTK
iuA libwebkitgtk-6.0-4              - Web content engine library for GTK
i   pandoc                          - general markup converter
```

Figure 5: Identifying non-Debian packages.

```
/usr/share/man/man1/zstdgrep.1.gz                                    OK
/usr/share/man/man1/zstdless.1.gz                                    OK
/usr/share/man/man1/zstdmt.1.gz                                      OK
bb@ilvarness:~$ debsums --changed
debsums: can't open blueman file /usr/share/polkit-1/rules.d/blueman.rules (Permission denied
)
debsums: can't open blueman file /var/lib/polkit-1/localauthority/10-vendor.d/org.blueman.pkl
a (Permission denied)
debsums: can't open bolt file /usr/share/polkit-1/rules.d/org.freedesktop.bolt.rules (Permiss
ion denied)
debsums: can't open cups-browsed file /usr/lib/cups/backend/implicitclass (Permission denied)
debsums: can't open cups-filters file /usr/lib/cups/backend/cups-brf (Permission denied)
debsums: can't open flatpak file /usr/share/polkit-1/rules.d/org.freedesktop.Flatpak.rules (P
ermission denied)
debsums: can't open flatpak file /var/lib/polkit-1/localauthority/10-vendor.d/org.freedesktop
```

Figure 6: Although `debsum` cannot run broken packages, it does list them.

`md5sum` program). The file starts with a summary of the repository, such as

```
Origin: Debian
Label: Debian
Suite: stable
Version: 12.8
Codename: bookworm
```

Using this file, you can generate a list of installed packages (Figure 5) that are not from an official Debian repository with

```
aptitude search '~S ~i !~ODebian !~o'
```

Replace `search` with `purge` and all non-Debian packages will be removed. To remove selected passages, use `apt-get purge` followed by a space-separated list of package names.

Dealing with Broken Packages

You may be familiar with broken packages from attempts at installing them. However, fixing broken packages during installation is a problem in its own right. The concern here is with packages that are broken when files are overwritten, usually by installing newer versions. These broken packages can be quickly located with the command `debsums --changed` (Figure 6). Note that `debsums` must be run from an ordinary account because it may refer to graphical applications that root is not set to run. Because the package is broken, `debsums` cannot be run from an ordinary account either, only identified. Each broken package can then be located with

```
dpkg --search PATH-TO-PACKAGE
```

and reinstalled.

Establishing a Routine

These cleanup tips can be used individually. Better yet, though, is to run them as part of a regular maintainance routine. Otherwise, you may experience unexpected memory shortages, resulting in such problems as `/var/cache/apt/archives/` being full and blocking all package installations and removals. Whether this scenario happens or how often depends on the size of your filesystem. However, when it does, you will have no choice except to delete files, resulting in a forced cleanup. If you do this housekeeping regularly, you are less likely to be forced to delete files in a panic that could encourage rash decisions and cause even more problems. ■■■

Info

[1] Raphaël Hertzog's blog: *https://raphaelhertzog.com*

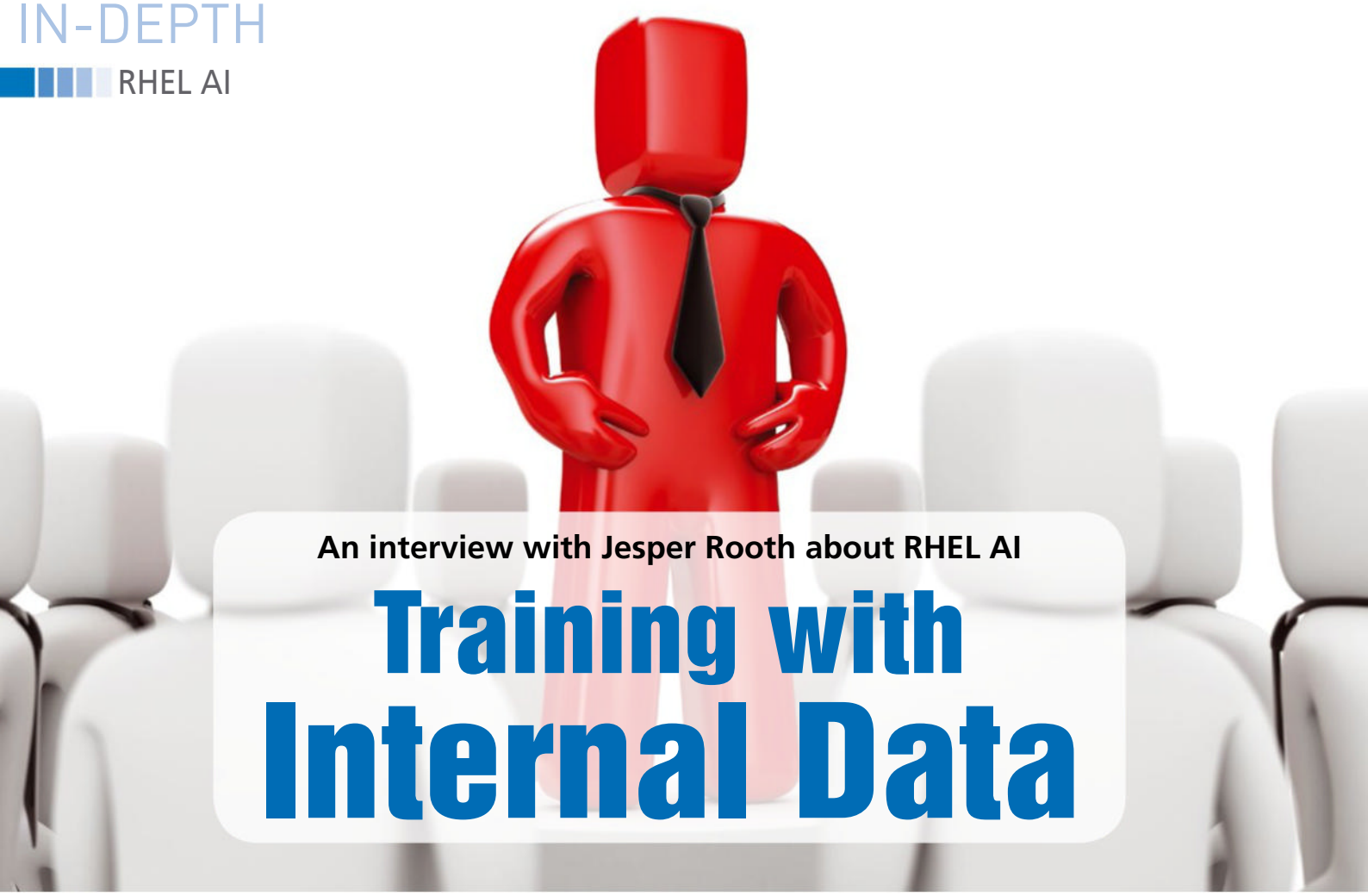
[2] Debian package information: *https://www.debian.org/distrib/packages*

[3] Package information on an installed system: *https://www.debian.org/distrib/packages*

[4] Cruft on the Debian wiki: *https://wiki.debian.org/Cruft*

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 cofounder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.



An interview with Jesper Rooth about RHEL AI

Training with Internal Data

Jesper Rooth discusses Red Hat's new AI platform solution as well as their future AI plans. *By Jens-Christoph Brendel*

Red Hat recently introduced Red Hat Enterprise Linux (RHEL) AI, a platform that enables the seamless development, testing, and deployment of enterprise applications using large language models (LLMs) from the IBM Granite family.

We spoke to Jesper Rooth, EMEA Linux Platform Lead at Red Hat, about Red Hat's AI plans.

Linux Magazine: What technical features distinguish the RHEL AI platform from standard servers?

Jesper Rooth, Red Hat EMEA Linux Platform Lead, is responsible for RHEL's platform strategies in the EMEA region. His responsibilities include working with customers and partners at an executive level to drive the adoption of RHEL across multiple industries. Rooth's career in Linux began in the late 1990s when he founded an open source, Linux consulting company in Sweden. Almost 20 years ago, Rooth joined Red Hat as a Linux consultant. He has held various positions at Red Hat, including head of the Scandinavian technology product sales and service team and EMEA sales manager for telecom technology products.

Jesper Rooth: RHEL AI is a base modeling platform that enables the development, testing, and deployment of enterprise applications using large Granite family language models. RHEL AI relies on a new feature in RHEL known as Image mode. Image mode is a deployment method that leverages the power of containers to integrate the various aspects of IT management into a single workflow. This makes Linux portable, scalable, and AI-enabled. On this basis, we can build an immutable RHEL installation containing only the software stack and the kernel to support generative LLM AI workloads. Basically, it's the same RHEL that users are used to, just wrapped with the immutability of Image mode.

LM: Is RHEL AI a tailored solution for the training phase of the model only, or can it also be used to advantage later when the trained model is run in an application?

JR: RHEL AI can also be used as a first-class solution for deployment/inferencing, as we provide vLLM as the inferencing engine.

LM: Who is RHEL AI aimed at? Is it intended for small and medium enterprises or more of an option for large corporations?

JR: RHEL AI is aimed at users who are looking for an easy-to-use, appliance-like solution for training their own models. RHEL AI is a single server and supports up to eight NVIDIA H100 GPUs. The target audience is organizations looking to inject their own confidential data directly into the model for specific use cases. Providing a consistent base model platform that is closer to an organization's data is critical to supporting AI strategies in production operations. As an extension of Red Hat's hybrid cloud portfolio, RHEL AI will cover almost every possible enterprise environment, from on-premise data centers, through edge environments, to the public cloud.

LM: What share of AI does Red Hat estimate that users will actually run on their own platforms in the near future, instead of consuming AI, say, in the form of a service?

JR: Red Hat believes that the future of AI must be open source and that organizations will generate more value from smaller, application-specific models trained with the organization's own data. The problem with using larger models – with more than a trillion parameters – is that the model knows everything, maybe even the DNA sequence of a spider. But that knowledge isn't very useful if a company wants to create a chatbot for its customer service agents or help customers solve a problem with its product. With RHEL AI and InstructLab, we are making the ability to train and run your own models more accessible and less expensive. InstructLab is an open source project for improving LLMs. Launched by IBM and Red Hat, this community project enables simplified experimentation with generative AI models and optimized model adaptation.

LM: Is this improvement of LLMs using InstructLab something that builds on a general understanding of the world and language of a trained base model in order to then add domain-specific knowledge? If so, how does this differ from Retrieval-Augmented Generation (RAG)?

JR: Yes, RHEL AI's approach is to improve a trained language model by adding domain-specific knowledge directly to the model itself. This is different than RAG, which is based on retrieving external information during use. By injecting knowledge into the model, RHEL AI reduces the complexity and cost of maintaining an external retrieval system, making it more efficient for domain-specific tasks.

LM: Are there already specific projects that customers have implemented or are looking to implement with RHEL AI?

JR: Yes, we are seeing huge demand for specific use cases for generative AI applications that contain an organization's confidential data directly in the model. If I had to pick just one use case, I would say RHEL AI and the Granite Code models that come with it as a local programming wizard for software engineers.

LM: Is Red Hat planning AI support beyond the infrastructure area, for example, in the form of its own applications? If so, in what form?

JR: Yes, we are currently offering our own models through the Granite family of models, which are open source LLMs licensed under Apache 2.0 with full transparency on the training datasets. We will also continue to provide additional AI capabilities, such as the Red Hat Ansible Lightspeed Generative AI service with the IBM watsonx Code Assistant or the Red Hat OpenShift Lightspeed AI-based virtual assistant.

LM: Thank you very much for this insight into Red Hat's plans for artificial intelligence!

Developing a mailbot script

Address Catcher

A Python script that captures email addresses will help you understand how bots analyze and extract data from the web. *By Andrea Ciarrocchi*

Bots crawl around constantly on the Internet, capturing information from public websites for later processing. Although the science of bot design has become quite advanced, the basic steps for capturing data from an HTML page are quite simple. This article describes an example script that extracts email addresses. The script even provides the option to extend the search to the URLs found on the target page. Rolling your own bot will help you build a deeper understanding of privacy defense and cybersecurity.

Setting Up the Environment

I recommend setting up an integrated development environment, like Visual Studio (VS) Code for Python programming, and having a basic understanding of the language. You can download VS Code from the VS Code website [1]. On Ubuntu, an easy way to install the application is by downloading the `.deb` package, right-clicking the file, and selecting the `Install` option. Alternatively, you can search for "vscode" in the App Center and click the `Install` button. If you prefer using the terminal, the VS Code website [2] provides detailed instructions for any Linux distribution. I also suggest adding Python development extensions, including Pylance and the Python Debugger.

Listing 1: Importing Modules

```
01 import urllib
02 import urllib.request
03 import re
04 import asyncio
05 import argparse
06 from alive_progress import alive_bar
07 import sys
```

The Script

The full text of the `mailbot.py` script is available on the *Linux Magazine* website [3]. Listing 1 shows the beginning of the script where I import the modules I will need to manage communications via the HTTP protocol, search for string patterns using regular expressions, implement asynchronous functions, manage script input arguments, and show a progress bar to track process advancement. The `alive_progress` module is not part of the standard library, so I have to install it with the following command:

```
pip install alive-progress
```

Listing 2 first defines the asynchronous function `ExtractURLs(myUrl)`, which takes the web address to be scanned as a parameter and returns a list of URLs. The code then analyzes each entry to search for email addresses in case the user has requested recursive scanning. The elements to be added to the list are selected and extracted using a regular expression, defined in the variable `regex` and used as a parameter in the `findall` function. The `findall` function returns all occurrences that match the format I have set. I store these occurrences in the list `t`, which is finally returned as the result of the `ExtractURLs(myUrl)` function.

Similarly, I then define the asynchronous function `ExtractMails(myUrl)`, which returns a list of email addresses found in the string `myUrl`, based on another regular expression that I have defined. Both functions described above are enclosed in a `try, except, finally` construct. In case of an error, the script does not perform any operations, in order to avoid premature termination or producing a final output in a nonstandard and thus unusable format. Regardless of the outcome, both functions return a list, containing web addresses and email addresses, respectively. The regular expressions used for extracting URLs and email addresses have been adapted from discussions on the Stack Overflow forum [4, 5]. I could refine `ExtractURLs(myUrl)` and `ExtractMails(myUrl)` to ensure they return a list strictly populated with valid values; however, I prefer to prioritize sensitivity over specificity in the extraction process. In other words, I choose to return a broader list of addresses that includes all available ones, rather than a

Listing 2: Defining Asynchronous Functions

```
01 async def ExtractURLs(myUrl):
02   try:
03     t=[]
04     regex="(?P<url>https?://[^\\s'\"]+)"
05     t=re.findall(regex, myUrl)
06   except:
07     pass
08   finally:
09     return t
10
11 async def ExtractMails(myUrl):
12   try:
13     q=[]
14     regex="[\\w\\.-]+@[\\w\\.-]+\\.\\w+"
15     u=re.findall(regex, myUrl)
16   except:
17     pass
18   finally:
19     return u
```

Listing 3: Processing Script Parameters

```
01 try:
02   parser=argparse.ArgumentParser()
03   parser.add_argument("url", help="Url to analyze", type=str)
04   parser.add_argument("output", help="Output file", type=str)
05   parser.add_argument("-r", "--recursive", help="Recursive search",
       action="store_true")
06   args=parser.parse_args()
07
08   response = urllib.request.urlopen(args.url)
09   webContent = response.read().decode('UTF-8')
```

shorter list of likely valid addresses that may exclude some equally valid entries. The rationale behind this choice is based on the fact that sending emails is a quick and low-cost operation. This approach allows me to maximize the number of users reached at the cost of a small percentage of undelivered emails.

In Listing 3 I process the script parameters, two of which are mandatory: the web address to analyze, `url`, and the name of the output file, `output`. The third parameter, `-r`, is optional and extends the search for email addresses to the URLs contained within the web page defined by the first parameter. I implement only one level of recursion, because the number of addresses involved in the search (and consequently the required resources) would grow exponentially with each iteration. Next, I open a connection to the web page specified in the `url` argument and assign its source code to the variable `webContent`.

Listing 4 scans the email addresses in the string I just obtained as a result of `ExtractURLs(myUrl)` function. At this point, I check whether the user has requested a recursive search. If not, the operation is completed, and I print the list variable `z` (where I have stored the found email addresses) to the output file. Otherwise, I perform a search for all URLs contained in the `webContent` string, and, for each of them, I call the `ExtractMails(myUrl)` function again. I perform the searches using asynchronous functions to ensure that the email extraction process occurs in an orderly fashion and only after the URL search has been completed. Specifically, the list `v` contains all the URLs extracted from `webContent` as its elements. I loop through the list `v`, calling the `ExtractMails(myUrl)` function for each element. The variable `k` contains the email addresses extracted from the current element of the `v` list. If the length of `k` is greater than zero (i.e., if at least one address is found), I add it to the list `z`, which holds all the email addresses extracted so far. At the same time, I provide the user with a visual progress indicator, using a real-time updating progress bar, which takes a value between zero and the length of the `v` list, corresponding to the number of URLs to analyze. The current value of the bar is equal to `i`, a counter that tracks the iterations through the URL list. The bar is updated at each iteration by calling the `bar()` function.

I check if any email addresses have been extracted and, if not, notify the user with the message "No mail addresses were found" (Figure 1). If the check is positive, I append a semicolon to the last element of the `z` list. Next, I open the file specified in the `args.output` argument and write all the elements of the `z` list to it, each separated by a semicolon and a space. Finally, I inform the user about the number of email addresses found. The output file will thus contain a sequence of addresses already properly formatted for use as a recipient list for an email.

Use Case Example

The script should be executed with the following syntax:

```
python mailbot.py website_name
  output_file [-r]
```

Instead of `website_name`, you should insert the full address of the page, including the "http://" or "https://" prefix. `output_file` refers to the text file where you want to save the list of found email addresses. Finally, the optional parameter `-r` enables a recursive search. Therefore, to recursively extract email addresses from *www.mysite.org* and save

Listing 4: Scanning Email Addresses

```
01   z=asyncio.run(ExtractMails(webContent))
02
03   if(args.recursive==True):
04     v=asyncio.run(ExtractURLs(webContent))
05     with alive_bar(len(v)) as bar:
06       for i in range(len(v)):
07         k=asyncio.run(ExtractMails(v[i]))
08         if(len(k)!=0):
09           z.extend(k)
10         bar()
11
12   if(len(z)>0):
13     z[-1] = z[-1] + ";"
14     with open(args.output, "w") as f:
15       print(*z, sep="; ", file=f)
16     print("Found " + str(len(z)) + " mail addresses.")
17   else:
18     print("No mail addresses were found.")
19 except Exception as e:
20   print(e)
```

the corresponding list to `output.txt`, you just need to type the following:

```
python mailbot.py ⮐
  https://www.mysite.org output.txt -r
```

A progress bar indicates the real-time progress of the search process. Figure 2 shows the appearance of the terminal while the operation is in progress. If the `-r` parameter is not specified, the search is performed only on the page provided as the first parameter of the script. The output file will contain a list of extracted email addresses, separated by a semicolon and a space. This way, the resulting string can be used directly as a recipient list without further processing.

Conclusion

Given how easy it is to automatically extract data from the web, I believe it is preferable to use contact forms rather than publicly displaying your email address. Alternatively, it is advisable to format the email in a way that avoids bot detection, such as *myaddress at mydomain dot com*. ■■■

Author

Andrea Ciarrocchi (*andreaciarrocchi.altervista.org*) is a technology enthusiast.

Info

[1] VS Code: *https://code.visualstudio.com/download*

[2] Installing from the terminal: *https://code.visualstudio.com/docs/setup/linux*

[3] Code for this article: *https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z*

[4] "Extracting a URL in Python," Stack Overflow: *https://stackoverflow.com/questions/839994/extracting-a-url-in-python*

[5] "Extract email sub-strings from large document," Stack Overflow: *https://stackoverflow.com/questions/17681670/extract-email-sub-strings-from-large-document*

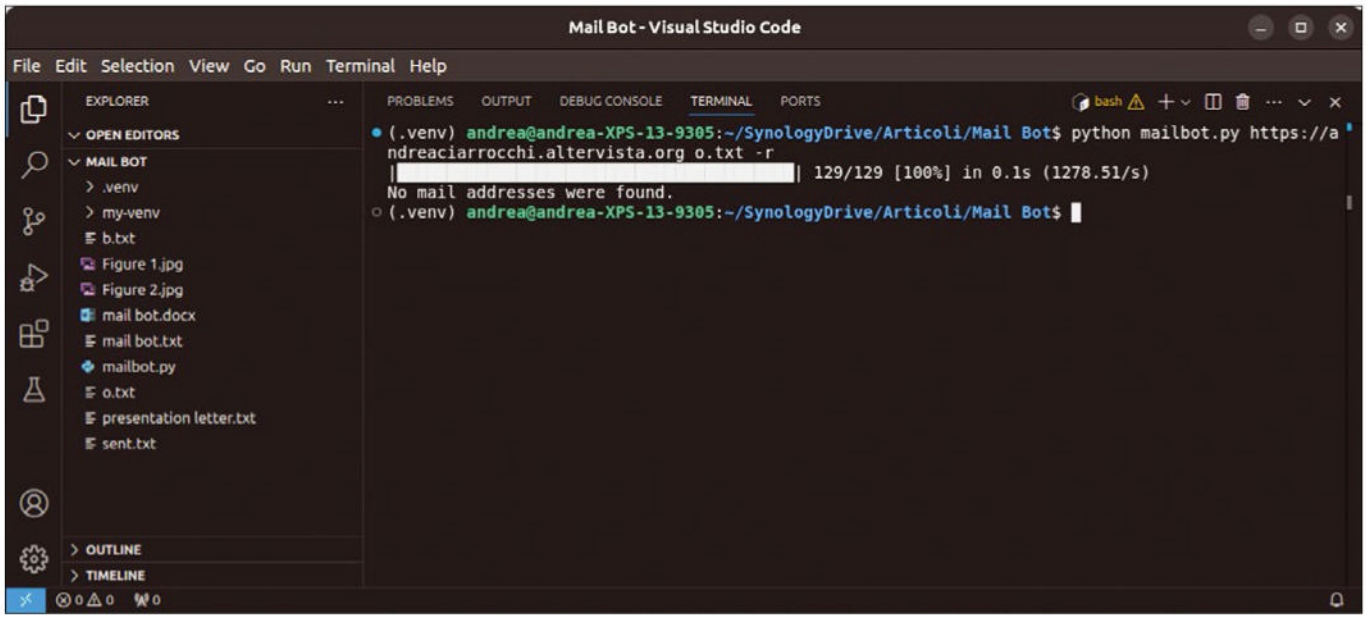


Figure 1: If no mail addresses are extracted, the mailbot issues a "No mail addresses were found" message.

Figure 2: Mailbot script at work in the terminal.

Hone Your Skills
– with –
Special Issues!

Get to know Shell, LibreOffice, Linux, and more from our Special Issues library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

Filter photos the grep way with the photogrep GUI tool

Free Selection

The photogrep GUI tool, built using Go and Fyne, is designed to filter photos just like the grep tool filters file names from a pipe. *By Mike Schilli*

The idea of pipes is one of the most powerful and versatile functions of the Unix command line. A pipe delivers the output from one tool into the input of the next, which, in turn, extracts interesting parts and passes those on to the next section of the pipeline to a further processing step. The fact that this invention never won a Nobel Prize is quite simply scandalous.

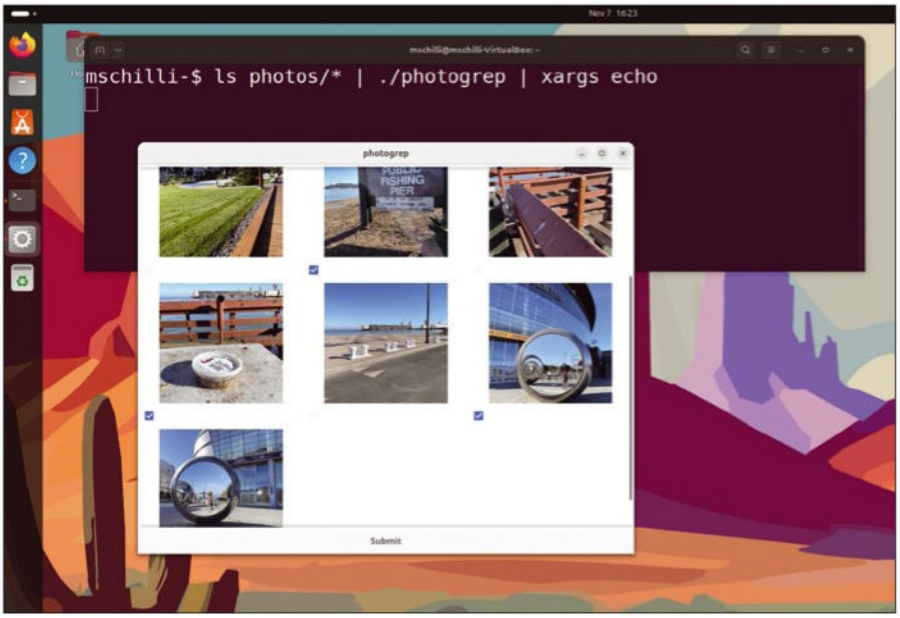
The grep command-line tool often plays a key role in such pipe processing, because it filters out unwanted entries between two connected pipe ends, keeping the good bits and dumping the bad ones into a black hole. When it comes to filtering strings, text replacement or regular expressions are massively useful, but what criteria can you use to sort photos? Which ones are beautiful and which ones are blurred, have a color cast, or simply have bad composition?

This kind of decision is (still) best made by a human being. The photogrep tool I will be discussing in this month's column fetches the photos from the stdin pipe and displays them on screen where you can select one or more of them with a mouse click. As soon as you press the *Submit* button, photogrep writes the names of the selected images to the standard output – hey presto, grep with photos.

Figure 1 shows photogrep in action, with a pipe whose first command lists all the image files in the `photos/` directory. The tool grabs the images from the standard input and displays them as thumbnails in a three-column matrix. You can now click on the images to pick them. A blue box with a white check mark appears for each selected photo.

To inspect the details of one of the photos, right-click on the image. A new viewer window then appears with the detailed view (Figure 2); you can move the content around like a map on Google Maps by holding down the mouse button and dragging. The detailed view disappears if you click on the window's close symbol.

Once you have selected the desired photos, clicking on *Submit* in the main window closes the tool. Shortly before



Lead Image © Venkova, 123RF.com

Figure 1: The photogrep tool helps you select photos.

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.

the program quits, photogrep sends the selected images' paths to its standard output, from which the next section of the pipe takes them in. In Figure 3, this is `xargs echo`, but it could be followed by arbitrary commands that automatically process the files arriving at their standard input, like moving or copying them somewhere.

Two Kinds of Input

Conjuring up a GUI on the desktop, with buttons that you can press and photos you can admire, is clearly a task for the Fyne framework with Go. The code can be compiled for all kinds of platforms, not only for Windows and Linux derivatives, but also for macOS, Android, and iOS.

Listing 1 shows the main program `main`, which first checks which way the files are coming in. Are the paths being sent line by line via the standard input, or has the user referenced them as command-line arguments? Just like grep, photogrep can handle both forms of input.

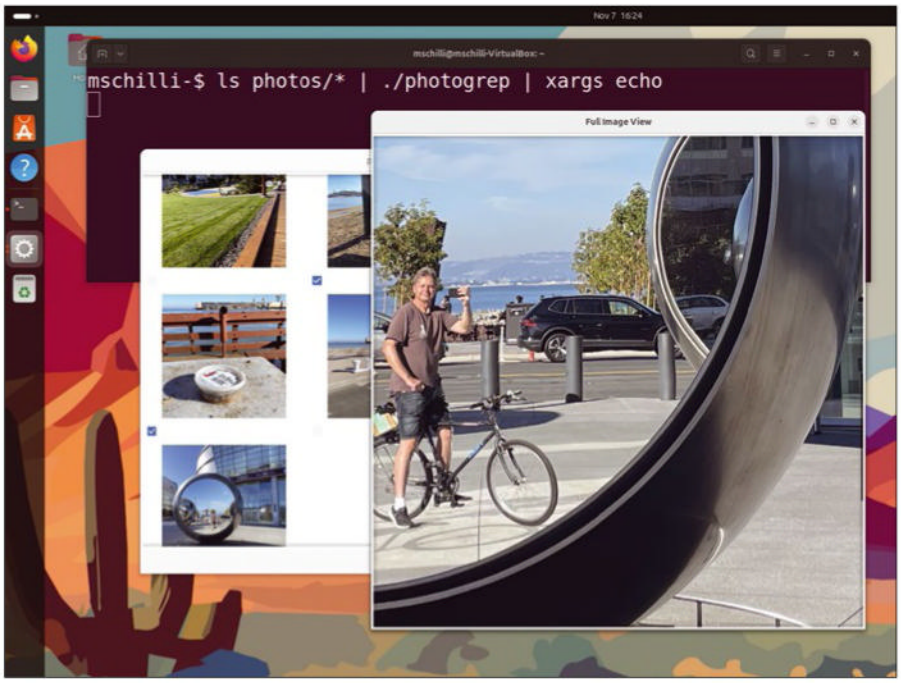


Figure 2: Right-clicking on a photo shows a movable section of the large original image.

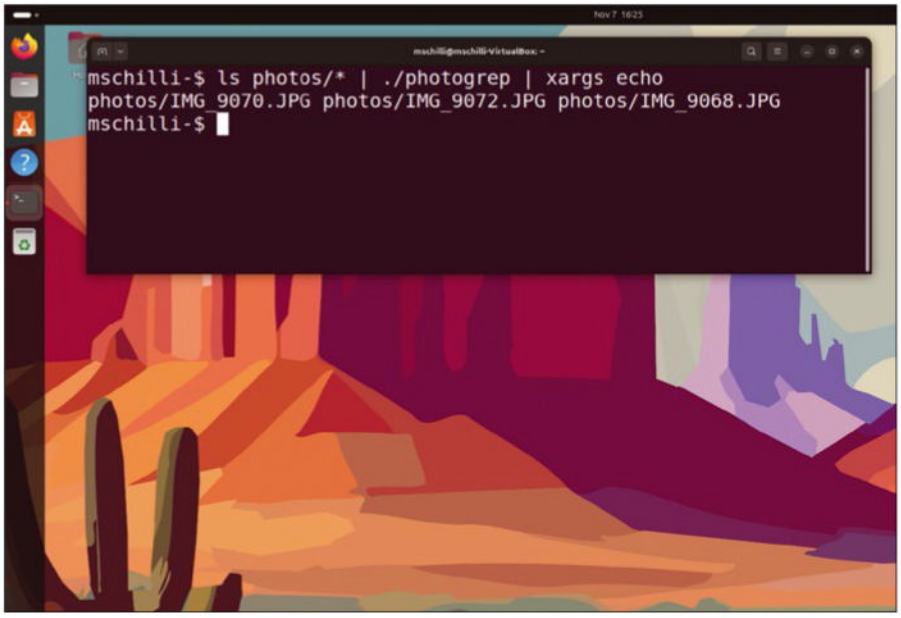


Figure 3: Clicking on *Submit* closes the tool and outputs the selected file paths.

Line 27 then calls `app.New()` to open a new app with the Fyne framework. After this, `NewWindow()` takes care of the associated GUI window. The photos displayed in the upper part look like a contact sheet, as you call it in analog photography, so that's what I called the corresponding widget. Its constructor, `NewCsheet()`, provides a reference to its data structure in line 29.

Calling the `MakeGrid()` function with the paths to the files to be displayed as parameters in line 30 displays the photos as a three-column matrix in a scrolling container. After all, the window has to fit in the limited desktop space and has a fixed size because of this. If more photos come in than the contact sheet widget can display, the window manager adds a vertical scrollbar that can be dragged to reveal the photos further down the page.

Line 31 defines a minimum fixed size of the contact sheet widget. This is crucial; otherwise, the scroll box would shrink to the smallest possible size. This often causes confusion when you're developing with Fyne, because suddenly you have an invisible 0x0 pixel widget on the desktop and are scratching your head wondering why it hasn't come up.

The app's main window now holds two components at the first level: the image gallery at the top and the *Submit* button at the bottom. The `NewButton()` function creates the button in line 33 and assigns a callback function to it. This function uses `cs.selected()` to list all user-selected photos on the contact sheet, iterates over their paths in a `for` loop, and calls `Println()` on each item to push the image paths out to the tool's standard output, line by line.

Remember Contact Sheets?

The constructor of the contact sheet widget, `NewCsheet()`, starts in line 19 of Listing 2. It returns a pointer to a structure in which it stores a reference to the app so that the code can later open a viewer window at the top level. Besides this, the struct's `selected` field type contains a hash map that flags each selected photo with a true `bool` entry.

Loading a dozen smartphone photos, each measuring 4000x3000 pixels, and displaying them as thumbnails takes a few seconds, even on fast hardware. This is why `MakeGrid()` calls `go func` to open a concurrent goroutine in line 30. It

reads in the photos to be displayed, scales them down, and draws them in the GUI. In the meantime, the main thread can return to the main program in line 37, bringing up the GUI with an empty contact sheet and updating the individual slots as they become available.

Staying Alive

If there is an unwritten law for visually appealing GUI applications, it's that GUIs can't hang and appear to be dead. The user always has to be able to click and trigger some kind of action. If an app hangs because the CPU is doing something other than processing events, you can be sure of a one-star rating in the App Store. Because the goroutine continues to run even though `MakeGrid()` has already finished, it gradually adds the finished files to the viewport courtesy of `grid.Add()`. Meanwhile, the GUI's event handler continues to respond to user requests in the predefined and natural way.

A smartphone photo can easily measure 25MB, and it would be extremely wasteful to display such a huge image in the GUI in a small 200x200 pixel peephole. The CPU would raster the full-resolution image and send it to the computer's GPU (if available). The GPU would then have to scale it down to the size to be displayed, which would slow down even the fastest hardware to creep speed.

Instead, Listing 2 draws in the *imaging* package from GitHub; it leverages the power of the Lanczos algorithm in the `Thumbnail()` function to scale down the images beforehand. The Fyne `NewImageFromImage()` function then quickly converts the thumbnail pixels into the GUI format. `newClickImage()` in line 59 then creates a clickable image. The function expects two callbacks, which define what happens when the user clicks the left or right mouse button while the pointer is hovering over the image.

Intuitive for Spoiled Users

If you think back to the first desktop designs, mouse operations were pretty awkward compared to the intuitive tap-and-swipe experience on today's tablets or smartphones. Schlepping the mouse to a checkbox below a photo just to click on it somehow feels like 1995. That's why line 61 in Listing 2 lets you simply left-click on the image to check or uncheck the selection box below it. Like magic!

The second callback starting in line 63 defines the response to a right mouse click on a photo. Line 64 opens a separate application window, while `Open()` from the imaging library displays the image with the orientation corrected courtesy of the `AutoOrientation` option being in effect. You might remember this from previous columns: Many smartphones save images rotated with the Exif header set to define the true orientation, but unfortunately many applications ignore this and display the image at odd angles. Line 71 then sends the 12 million pixels of the full resolution to the viewing window courtesy of the `NewPan()` function.

To support quick scrolling through oversized photos, Fyne offers `container.Scroll` areas with scrollbars at the sides. But locating these with the mouse pointer, clicking on them, and then moving them is just too much to ask of users in 2024. Instead, today's users expect to be able to drag the mouse to move the photo around in a flexible frame, just like in Google Maps – this is known as panning.

But let's talk about the clickable photos on the contact sheet first. Fyne defines a series of graphical primitives such as `canvas.Image`, which do not process mouse clicks for performance reasons. However, if you need a widget that is a little out of the ordinary, for example, a clickable photo, you can easily do this as shown in Listing 3. A construct

Listing 1: photogrep.go

```
01 package main
02
03 import (
04   "bufio"
05   "flag"
06   "fmt"
07   "fyne.io/fyne/v2"
08   "fyne.io/fyne/v2/app"
09   "fyne.io/fyne/v2/container"
10   "fyne.io/fyne/v2/widget"
11   "os"
12   "strings"
13 )
14
15 func main() {
16   flag.Parse()
17   var fileNames []string
18   if flag.NArg() != 0 {
19     fileNames = flag.Args()
20   } else {
21     scanner := bufio.NewScanner(os.Stdin)
22     for scanner.Scan() {
23       fileNames = append(fileNames, strings.
           TrimSpace(scanner.Text()))
24     }
25   }
26
27   app := app.New()
28   w := app.NewWindow("photogrep")
29   cs := NewCsheet(app)
30   grid := cs.MakeGrid(fileNames)
31   grid.SetMinSize(fyne.NewSize(800, 600))
32
33   submitButton := widget.NewButton("Submit", func() {
34     for fileName := range cs.selected {
35       fmt.Println(fileName)
36     }
37     w.Close()
38   })
39
40   content := container.NewVBox(grid, submitButton)
41   w.SetContent(content)
42   w.ShowAndRun()
43 }
```

based on `widget.BaseWidget` (line 10), containing a pointer to the image to be displayed, does the trick here. It also defines the `cbleft()` and `cbright()` callbacks for actions with the left and right mouse button.

Left Click, Right Click, Go!

Fyne's scope is not restricted to running GUIs on the desktop; it also supports apps on mobile devices. This means that it's a bit awkward to define what happens after a left or right click with the mouse. As we all know, cell phones do not have a mouse – instead, the user taps, holds, or swipes. Fyne categorizes the first two events as `Tapped()` and `TappedSecondary()`. In a desktop context, these define left and right mouse clicks. Listing 3 defines a callback for both, which the caller later passes to the exported `newClickImage()` function.

I still need to provide instructions for Fyne on how to display my new custom widget. Line 25 of Listing 3 simply lets the widget's mandatory `CreateRenderer()` function use the predefined basic `NewSimpleRenderer` for my images. But what about the viewer window, which pops up after a right mouse click? How do I tell Fyne to support dragging the mouse through the different areas of a high-resolution image, just like in Google Maps? To let users do this, Listing 4 creates another custom widget named `Pan` and pulls out all the stops.

Listing 2: csheet.go

```
01 package main
02
03 import (
04   "fyne.io/fyne/v2"
05   "fyne.io/fyne/v2/canvas"
06   con "fyne.io/fyne/v2/container"
07   "fyne.io/fyne/v2/widget"
08   "github.com/disintegration/imaging"
09 )
10
11 const ThumbSize = float32(200)
12 const ViewSize = float32(800)
13
14 type csheet struct {
15   selected map[string]bool
16   app      fyne.App
17 }
18
19 func NewCsheet(app fyne.App) *csheet {
20   return &csheet{
21     selected: map[string]bool{},
22     app:      app,
23   }
24 }
25
26 func (cs *csheet) MakeGrid(fileNames []string)
      *con.Scroll {
27   grid := con.NewGridWithColumns(3)
28   scroll := con.NewScroll(grid)
29
30   go func() {
31     for _, fileName := range fileNames {
32       pick := cs.newPick(fileName)
33       grid.Add(pick)
34     }
35   }()
36
37   return scroll
38 }
39
40 func (cs *csheet) newPick(fileName string)
      *fyne.Container {
41   img, err := imaging.Open(fileName, imaging.
       AutoOrientation(true))
42   if err != nil {
43     panic(err)
44   }
45   thumbnail := imaging.Thumbnail(img, int(ThumbSize),
       int(ThumbSize), imaging.Lanczos)
46
47   image := canvas.NewImageFromImage(thumbnail)
48   image.FillMode = canvas.ImageFillContain
49   image.SetMinSize(fyne.NewSize(ThumbSize, ThumbSize))
50
51   check := widget.NewCheck("", func(checked bool) {
52     if checked {
53       cs.selected[fileName] = true
54     } else {
55       delete(cs.selected, fileName)
56     }
57   })
58
59   ci := newClickImage(image, func() {
60     // toggle
61     check.SetChecked(!check.Checked)
62   },
63     func() {
64       fullView := cs.app.NewWindow("Full Image View")
65       img, err := imaging.Open(fileName, imaging.
           AutoOrientation(true))
66       if err != nil {
67         panic(err)
68       }
69       fullImage := canvas.NewImageFromImage(img)
70       fullImage.FillMode = canvas.ImageFillOriginal
71       inspector := NewPan(fullImage)
72       fullView.SetContent(inspector.scroll)
73       fullView.Resize(fyne.NewSize(ViewSize, ViewSize))
74       inspector.Center()
75       fullView.Show()
76     })
77
78   image.Refresh()
79   return con.NewVBox(ci, check)
80 }
```

Listing 3: clickimg.go

```
01 package main
02
03 import (
04   "fyne.io/fyne/v2"
05   "fyne.io/fyne/v2/canvas"
06   "fyne.io/fyne/v2/widget"
07 )
08
09 type clickImage struct {
10   widget.BaseWidget
11   image   *canvas.Image
12   cbleft  func()
13   cbright func()
14 }
15
16 func newClickImage(img *canvas.Image, cbleft func(),
      cbright func()) *clickImage {
17   ci := &clickImage{}
18   ci.ExtendBaseWidget(ci)
19   ci.image = img
20   ci.cbleft = cbleft
21   ci.cbright = cbright
22   return ci
23 }
24
25 func (t *clickImage) CreateRenderer() fyne.WidgetRenderer {
26   return widget.NewSimpleRenderer(t.image)
27 }
28
29 func (t *clickImage) Tapped(_ *fyne.PointEvent) {
30   t.cbleft()
31 }
32
33 func (t *clickImage) TappedSecondary(_ *fyne.PointEvent) {
34   t.cbright()
35 }
```

Listing 4: pan.go

```
ackage main
02
03 import (
04   "fyne.io/fyne/v2"
05   "fyne.io/fyne/v2/canvas"
06   "fyne.io/fyne/v2/container"
07   "fyne.io/fyne/v2/widget"
08   "image/color"
09 )
10
11 type Pan struct {
12   widget.BaseWidget
13   image        *canvas.Image
14   scroll       *container.Scroll
15   scrollOffset fyne.Position
16 }
17
18 func NewPan(img *canvas.Image) *Pan {
19   di := &Pan{image: img}
20   di.ExtendBaseWidget(di)
21
22   scrollContent := container.NewMax(di)
23   scroll := container.NewScroll(scrollContent)
24   di.scroll = scroll
25   return di
26 }
27
28 func (di *Pan) CreateRenderer() fyne.WidgetRenderer {
29   return &panRenderer{di}
30 }
31
32 func (di *Pan) Dragged(e *fyne.DragEvent) {
33   di.scroll.Offset = di.scroll.Offset.SubtractXY(e.
       Dragged.DX, e.Dragged.DY)
34   di.scroll.Refresh()
35 }
36
37 func (di *Pan) Center() {
38   w, h := di.image.Size().Width, di.image.Size().Height
39   di.scroll.Offset = fyne.NewPos(float32(w)/2.0,
       float32(h)/2.0)
40   di.scroll.Refresh()
41 }
42
43 func (di *Pan) DragEnd() {
44 }
45
46 type panRenderer struct {
47   di *Pan
48 }
49
50 func (r *panRenderer) Layout(size fyne.Size) {
51   r.di.image.Resize(size)
52 }
53
54 func (r *panRenderer) MinSize() fyne.Size {
55   return r.di.image.MinSize()
56 }
57
58 func (r *panRenderer) Refresh() {
59   canvas.Refresh(r.di.image)
60 }
61
62 func (r *panRenderer) BackgroundColor() color.Color {
63   return color.Transparent
64 }
65
66 func (r *panRenderer) Objects() []fyne.CanvasObject {
67   return []fyne.CanvasObject{r.di.image}
68 }
69
70 func (r *panRenderer) Destroy() {}
```

The Pan Widget

Again, the `Pan` widget is based on `widget.BaseWidget` (line 12, Listing 4). It defines an attribute named `image`, which points to the photo to be displayed. This Google Maps look-alike needs a scroll container (`scroll` in line 14) and a marker for the current scroll position (`scrollOffset`). As I mentioned earlier, there is a scroll widget in Fyne that can load and display oversized photos, but navigating the areas of the image relies on clickable and draggable scrollbars, located to the right and below the display area.

The trick with `Pan` is that it lets you navigate the details of the photo by holding down the left mouse button and dragging. To let this happen, `Dragged()` receives the current mouse coordinates multiple times per second starting in line 32 as the user drags the mouse through the image while holding down the left button. Line 33 subtracts these delta values from the offset stored in the scroll widget and uses `Refresh()` to display the image area section at the new coordinates in the peephole.

Send or Dismiss?

Why subtract and not add? The first map implementations during the Internet stone age actually forced users to drag the mouse to point out the direction in which an app had to move the mouse pointer on the map. Google Maps reversed the direction. Today, it would be unthinkable to implement anything other than using a press-and-drag mouse action to indicate the direction in which to push the underlying map (or photo).

Now, when you call up the inspector window to look at the details of a thumbnail, it is highly unlikely that the top left-hand corner of the photo is what you want to see, so the image center is a better place to start panning. The `Center()` function starting in line 37 positions the peephole at coordinates that are exactly half the width and height of the image.

Because of the widget's more complex functionality, the `Pan` widget in Listing 4 does not use the simple image renderer, in contrast to Listing 3, but defines its own `panRenderer` starting in line 46. On request, this renderer needs to tell the Fyne widget manager how to scale up if space becomes available (`Layout()`), what the minimum size (`MinSize()`) is, or how to display the latest view on-screen (`Refresh()`). The custom widget falls back to the displayed photo widget's functions for this. Its `canvas.Image` implementation already provides everything necessary.

I was surprised to see that the panning actually works surprisingly quickly in the end. Hardware support is involved via the scroll container: No CPU in the world could calculate a section of an image that quickly.

Forging Tools

The usual rule of three from Listing 5 compiles all the listings in this month's column to create the `photogrep` binary. The executable will process photos delivered either via the standard input or as command-line arguments. If you have never worked with Fyne on Linux before, you will want to install *golang-go*, *build-essential*, *libgl1-mesa-dev*, and *xorg-dev* up front. Like grep, the new practical photogrep tool is something no toolbox should be without. ■■■

Listing 5: build.sh

```
$ go mod init photogrep
$ go mod tidy
$ go build photogrep.go clickimg.go csheet.go pan.go
```

Pattern-matching tools for chasing down malicious software

GATE SEARCH

The big antivirus companies offer a myriad of malware scanning utilities, but it is often difficult to see what they are really doing or to customize them for specific needs. Beyond the giants are a class of more versatile tools that let you choose the rulesets – and even write your own rules. *By Chris Binnie*

The words *virus* and *malware* are often used in the same breath, although they refer to different things. Malicious software (typically shortened to "malware") covers a vast array of unwelcome software threats that fall into multiple categories. The term *malware* essentially refers to any code designed to cause harm. A virus, on the other hand, is a type of malware that generally becomes active when a legitimate piece of software is executed. Like a real virus, a software virus has a way of replicating, allowing it to spread to any system it is capable of infecting.

In this article, I will look at software that allows anyone familiar with Linux and a willingness to learn to create rules for detecting and classifying malware.

YARA

YARA [1] is provided by VirusTotal [2] and is released under the permissive BSD 3-Clause "New" License, which means that you can use it for commercial purposes [3]. Google acquired VirusTotal in 2012 [4]. The name YARA apparently stands for "Yet Another Ridiculous Acronym"!

The YARA documentation [5] describes YARA as a tool that can "…create descriptions of malware families (or whatever you want to describe) based on textual or binary patterns." The YARA website [6] includes an impressive list of some of the companies using the tool, including Trend Micro, Kaspersky Lab, SonicWall, ESET, and Avast. The YARA project refers to its toolset as "The pattern matching swiss knife for malware researchers (and everyone else)."

You can install YARA via the package manager on several Linux distributions, but I will focus on Debian Linux derivatives (in this case, Ubuntu 22.04 "Jammy Jellyfish" installed in a VirtualBox virtual machine). YARA will also run on Mac and Windows platforms, and if you want the very latest version, you can compile it yourself [7].

The following command gets the installation process started:

```
$ apt install -y yara
```

Two new packages are installed, taking up less than half a megabyte of disk space. In addition to the *yara* package, the installer sets up the *libyara8* library.

The YARA documentation is detailed and easy to follow. The docs encourage you to write your first malware rule, which I will do right now:

```
$ echo "rule not_really_malware
  { condition: true }" > my_first_rule
```

The next step is to run YARA against this rule. The usual way to run YARA is to pass it a rule name and the name of a file you wish to scan for malware. I have copied the file `/etc/issue` into a temporary directory that I will run YARA in. `/etc/issue` is a file that gets called (unless you configure your system not to do so) when a user logs in, prior to the login prompt. In my case, the contents is simply:

```
Ubuntu 22.04.3 LTS \n \l
```

The `\n` option displays the hostname or node name, and the `\l` seems to insert the name of the current teletypewriter (TTY) interface name afterwards.

I am ready to run YARA for the first time, as shown here, with the rule name first and then the file to be scanned:

```
$ yara my_first_rule issue
not_really_malware issue
```

As you can see the rule with the filename `my_first_rule`, which will always report as true thanks to the condition, has declared a finding for the threat `not_really_malware` in our example test file `issue`.

Rules Are Rules

Each rule always starts with the word `rule` and then follows the popular YAML (YAML Ain't Markup Language) file format. Rules are then normally constructed with a strings definition section and then a condition section. Text is enclosed within double quotes (inverted

Reserved Rule Names (Identifiers)			
all	for	int8	strings
and	fullword	int8be	them
any	global	istartswith	true
ascii	import	matches	uint16
at	icontains	meta	uint16be
base64	iendswith	nocase	uint32
base64wide	iequals	none	uint32be
condition	in	not	uint8
contains	include	of	uint8be
endswith	int16	or	wide
entrypoint	int16be	private	xor
false	int32	rule	defined
filesize	int32be	startswith	

Figure 1: Reserved YARA keywords that aren't allowed as rule names (identifiers).

commas) and hexadecimal strings are surrounded by curly brackets.

The rule name is referred to as an identifier. Figure 1 shows reserved identifiers that cannot be used, thanks to the fact that YARA uses these as keywords in its own rule processing. Case-sensitive rule names cannot start with a number, but other than that, the name can consist of any alphanumeric character, as long as the name is less than 128 characters long.

Listing 1 shows an example of a typical, two-sectioned rule. As you might guess from the condition section in Listing 1, the rule will trigger and find a positive result if either the plain text string or the hexadecimal is matched during a file scan. Note the `or` keyword used in the condition. I'm sure you are starting to appreciate that YARA rules are clean and logical in their construction. It is possible to use global rules that you can pass to other rules, such as this example from the YARA documentation:

```
global rule SizeLimit
{
    condition:
        filesize < 2MB
}
```

Additionally, modules are supported – and some modules are bundled with the tool directly. Modules extend the main functionality that YARA provides and can create much more fine-grained rules.

Populating your rules with variables is also possible, and you can pull in and

Listing 1: Two-Sectioned Rule

```
rule two_main_sections
{
    strings:
        $some_text = "identifying malware text"
        $some_hexadecimal = { 44 B3 45 ED A2 14 C4 44 B3 45 ED A4 23 }
    condition:
        $some_text or $some_hexadecimal
}
```

```
22-04 yara # file my_first_rule
my_first_rule: ASCII text
22-04 yara # yarac my_first_rule compiled_my_first_rule
22-04 yara # file compiled_my_first_rule
compiled_my_first_rule: YARA 3.x compiled rule set development version 00
22-04 yara #
```

Figure 2: Compiling a rule in YARA with the `yarac` binary.

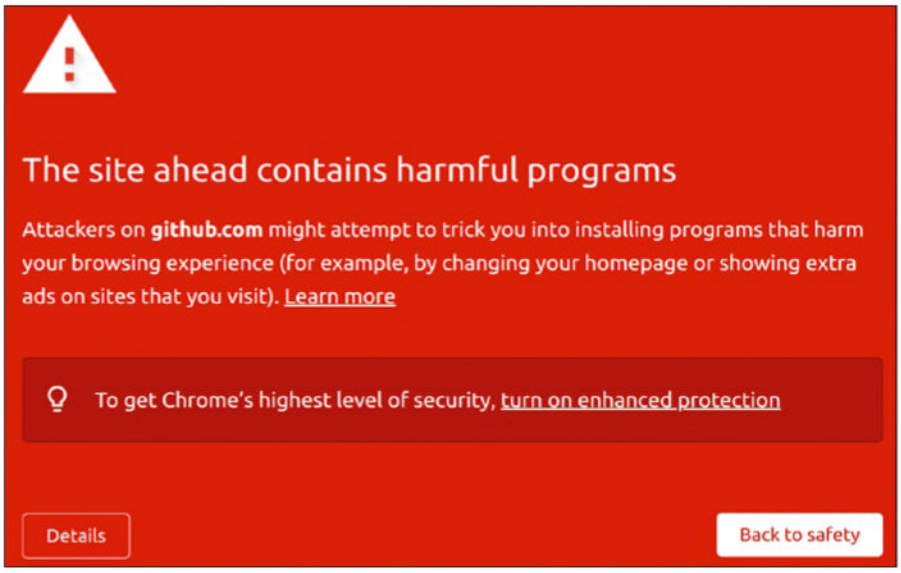


Figure 3: Google Chrome is doing its job and alerting about the presence of malware.

Listing 2: Matching mimikatz Malware

```
rule mimikatz_test
{
  meta:
     author = "fullsecurityengineer.com"
     description = "A rule for detecting ParrotSec mimikatz"
  strings:
     $s1 = {4d 5a} // Windows Executable File Magic Numbers
     $s2 = "benjamin@gentilkiwi.com0" ascii fullword
     $s3 = "$http://blog.gentilkiwi.com/mimikatz 0" ascii fullword
  condition:
     $s1 at 0 and $s2 and $s3
}
```

Listing 3: YARA Output

```
$ yara -r ~/mimikatz_test mimikatz/

mimikatz_test mimikatz//Win32/mimilove.exe
mimikatz_test mimikatz//Win32/mimilib.dll
mimikatz_test mimikatz//x64/mimilib.dll
mimikatz_test mimikatz//Win32/mimikatz.exe
mimikatz_test mimikatz//x64/mimikatz.exe
```

include files too if you wish. YARA follows much of the same syntax as the C language, so the format for multiple-line comments is as follows:

```
/*
    Comment for blocks of code
*/
```

And, for single line comments, you can use the following format:

```
// nothing to see here, move along
```

It might have already occurred to you that running YARA in an enterprise environment could mean iterating through tens of thousands of rules. Fret not – YARA lets you compile the rules, which makes them highly performant.

YARA bundles a binary called `yarac` that you can use to compile rules. Figure 2 shows the output from the `file` command after compiling `my_first_rule`.

You can load up compiled rules by using the `-C` option:

```
$ yara -C ➥
  compiled_my_first_rule issue
not_really_malware issue
```

Cats and Dogs

To show you YARA in action, the malware that I will test against is called `mimikatz`. Hosted on GitHub [8], `mimikatz` claims to be able to "extract plaintext passwords, hash, PIN code, and Kerberos tickets from memory."

As soon as I click the *mimikatz* link, Google Chrome panics with the page shown in Figure 3.

In Figure 3, you can see that Chrome's built-in security is working as hoped. (Opening the page in Firefox yields a similar warning.)

I found an excellent site called Full Security Engineer [9] with some very useful security information and used the rule on display there, as shown in Listing 2.

And, as if by magic, YARA displays multiple findings (Listing 3). The `-r` option recursively scans all files in a directory tree.

As should be obvious from the output, multiple files have been matched using some of the strings in the rule in Listing 2, specifically the following:

```
$s1 = {4d 5a} // Windows ➥
  Executable File Magic ➥
  Numbers
$s2 = "benjamin@gentilkiwi.com0" ➥
  ascii fullword
$s3 = "$http://blog.gentilkiwi.com/➥
  mimikatz 0" ascii fullword
```

Running the command `yara -help`, as you would expect, offers lots of other options. The `-s` switch, for instance, will print matching strings, which is ideal for debugging.

Having compiled the `mimikatz_test` rule, I can now pull together a recursive scan with pattern-matches, using the following command with the abbreviated output as shown in Figure 4:

```
$ yara -sCr ➥
  compiled_mimikatz_test ➥
  mimikatz/
```

Public Enemy

I found an intriguing GitHub repository with a seemingly endless number of rules. I decided to clone it, having spotted a file [10] mentioning Advanced Persistent Threats (APT) and a rule called `Dropper_DeploysMalwareVia-SideLoading`. I started with this command:

```
$ git clone https://github.com/➥
  Yara-Rules/rules.git
```

```
22-04 yara # yara -sCr compiled_mimikatz_test mimikatz/
mimikatz_test mimikatz//Win32/mimilove.exe
0x0:$s1: 4D 5A
0x68b9:$s2: benjamin@gentilkiwi.com0
0x7f33:$s2: benjamin@gentilkiwi.com0
0x6ddc:$s3: $http://blog.gentilkiwi.com/mimikatz 0
0x8479:$s3: $http://blog.gentilkiwi.com/mimikatz 0
mimikatz_test mimikatz//Win32/mimilib.dll
0x0:$s1: 4D 5A
0x7cb9:$s2: benjamin@gentilkiwi.com0
0x9333:$s2: benjamin@gentilkiwi.com0
0x81dc:$s3: $http://blog.gentilkiwi.com/mimikatz 0
0x9879:$s3: $http://blog.gentilkiwi.com/mimikatz 0
mimikatz_test mimikatz//x64/mimilib.dll
```

Figure 4: Lots of matched output, abbreviated.

```
22-04 rules # ls
antidebug_antivm              crypto_index.yar      email_index.yar       index.yar          malware_index.yar      README.md
antidebug_antivm_index.yar    cve_rules             exploit_kits          LICENSE            mobile_malware         utils
capabilities                  cve_rules_index.yar   exploit_kits_index.yar maldocs           mobile_malware_index.yar webshells
capabilities_index.yar        deprecated            index_gen.sh          maldocs_index.yar  packers                webshells_index.yar
crypto                        email                 index_w_mobile.yar    malware            packers_index.yar
22-04 rules #
```

Figure 5: Lots of useful rules in the Yara-Rules repository.

And, once inside the resulting `rules/` directory, I could see the directory listing shown in Figure 5.

Figure 5 shows some of the available rules, such as rules for email, crypto, and web shells. In the `exploit_kits` directory, the following files are intriguing:

```
EK_Angler.yar EK_BleedingLife.yar
EK_Eleonore.yar EK_Phoenix.yar
EK_ZeroAcces.yar  EK_Zeus.yar
EK_Blackhole.yar EK_Crimepack.yar
EK_Fragus.yar EK_Sakura.yar
EK_Zerox88.yar
```

Note that you'll need YARA version 3 or later to run these rules. Figure 6 shows what happens when you use the `capabilities/` directory rules against the poisoned `mimikatz/` directory (I have left off the `-s` output as it was very noisy).

According to the documentation for the ruleset, the `capabilities/` directory rules are "…to detect capabilities that do not fit into any of the other categories. They are useful to know for analysis but might not be malicious indicators on their own." You'll find a long list of curated rule repositories, as well as a useful YARA tutorial, on GitHub [11] [12].

Other Resources

I would be remiss not to mention that makers of YARA also provide a commercial tool [13] that permits malware analysis directly in a browser (Figure 7).

I used VirusTotal to analyze a few files. I tried the online option (not the local file-upload option) against an infected `mimikatz` executable that is available online [14]. As you can see in Figure 8, the online service successfully identified the file as a threat.

If you want to automate your own checks against the latest threat intelligence feeds, you can also hook into VirusTotal's API [15]. The functionality is impressive and already at a version 3 implementation. Table 1 shows some of the features.

See the VirusTotal website for more information on the difference between the public API and the premium API [16].

More, More, More

One commercial offering that incorporates YARA is called Thor, from Nextron Systems [17]. The community edition (known as Thor Lite [18]) is available for free (for noncommercial use) and comes with an extensive open source set of signatures (which was also included in the previous incarnation of Thor, which was called Loki [19]).

Both products are designed to provide the ability to scan files for Indicators of

```
22-04 rules # yara -r capabilities/* mimikatz/
win_token mimikatz//Win32/mimilib.dll
win_files_operation mimikatz//Win32/mimilove.exe
win_token mimikatz//x64/mimilib.dll
inject_thread mimikatz//Win32/mimikatz.exe
create_service mimikatz//Win32/mimikatz.exe
network_dns mimikatz//Win32/mimikatz.exe
cred_local mimikatz//Win32/mimikatz.exe
spreading_share mimikatz//Win32/mimikatz.exe
win_mutex mimikatz//Win32/mimikatz.exe
win_registry mimikatz//Win32/mimikatz.exe
win_token mimikatz//Win32/mimikatz.exe
win_files_operation mimikatz//Win32/mimikatz.exe
inject_thread mimikatz//x64/mimikatz.exe
create_service mimikatz//x64/mimikatz.exe
network_dns mimikatz//x64/mimikatz.exe
cred_local mimikatz//x64/mimikatz.exe
spreading_share mimikatz//x64/mimikatz.exe
win_mutex mimikatz//x64/mimikatz.exe
win_registry mimikatz//x64/mimikatz.exe
win_token mimikatz//x64/mimikatz.exe
win_files_operation mimikatz//x64/mimikatz.exe
22-04 rules #
```

Figure 6: We have several winners.

Figure 7: VirusTotal lets you upload files for analysis.

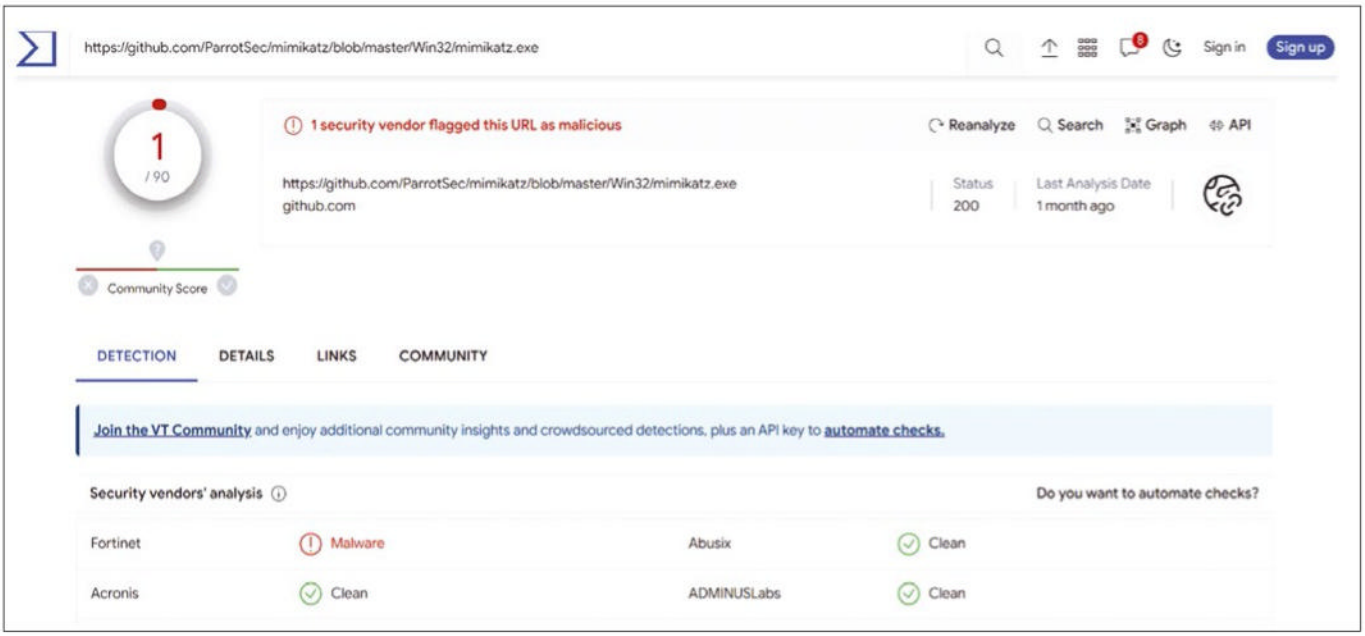


Figure 8: The URL contains malware.

Compromise (IoCs). IoCs provide precise details about attempted or successful security exploits, which are expertly coupled with the functionality that the inimitable YARA provides for spotting suspicious files.

The set of signatures used by Thor are touted as being top quality [20]. For a fee, you can significantly increase the number of signatures from that repository (run by Florian Roth, Head of R&D at Nextron Systems) if you opt to use the commercial Valhalla ruleset [21]. That page is worth a read; if you are part of an organization, it makes sense to sign up.

That said, the open source feed with Thor Lite (which requires registration) apparently contains over 4,000 YARA rules. Valhalla boasts over 17,000, plus 10,000 IoC patterns, plus the open source rules.

Improvements on Loki mean that Thor is much faster, as it is written in Go and as a result supports CPU throttling. Thor can also egress its logging to a number of sources, including syslog and even JSON over UDP/TCP.

Once you have subscribed to the newsletter and registered your details, you are promised "… an email with a THOR-Lite-generated personal license file (`*.lic`) and a download link for the THOR Lite ZIP package."

It is then a case of downloading your license, followed by the Zip file containing Thor Lite. The license lasts for a year, and then you will receive a new license. Unsubscribing from the newsletter will also end the download subscription.

As promised, the license file has a file extension of `.lic`. The *Download for Linux* button requires a EULA acceptance, and then a 36MB Zip file is downloaded. To get started, I decompress the file and then run the following command:

```
$ ./thor-lite-linux-64
```

Some welcome ASCII art appears (Figure 9), followed by lots of interesting output.

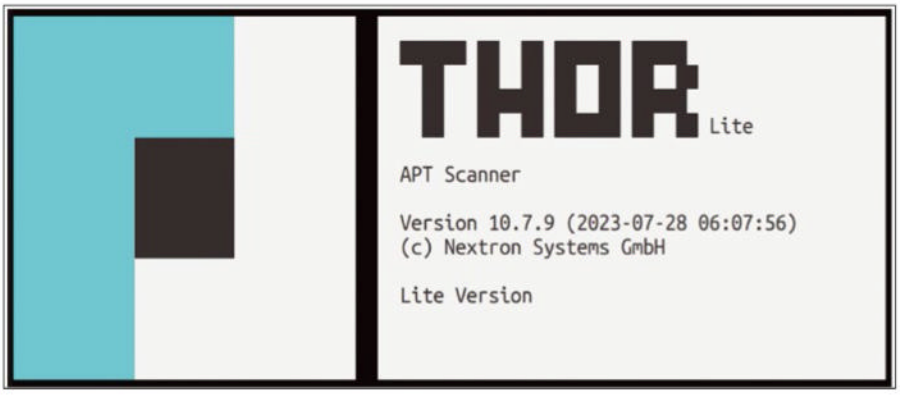


Figure 9: ASCII lettering courtesy of Thor.

Table 1: VirusTotal Features

Header	Description
Upload a file to scan	Scan a file via over 70 antivirus products and multiple other security tools.
Check by hash	Pass a MD5, SHA-1 or SHA-256 hash to identify precisely the file you are scanning.
Scan by URL	Scan a URL's contents with over 70 antivirus products and multiple other security tools.
Generate URL report	Create a useful report against a URL's scan.
Create a report by domain name	Generate analysis in a report by domain name.
Check by IP address	Create a useful report against an IP address.

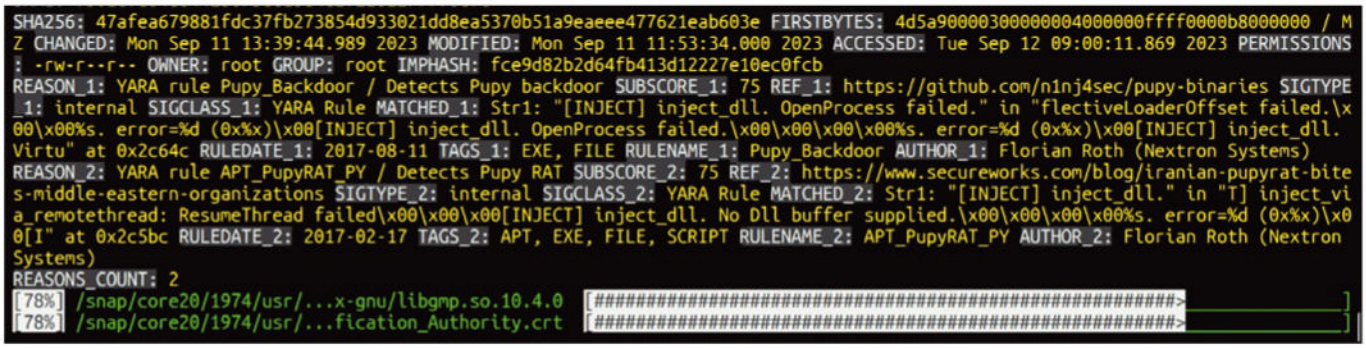


Figure 10: Thor is hammering home the need for ongoing security.

Figure 10 shows the continued output by the formidable Thor after it has run for five minutes or so.

I didn't expect Thor to be quite as thorough in the way it scrutinized my laptop's filesystem. It went off like a rocket, but surprisingly, the system load didn't increase massively.

Thor's output is multicolored and reveals intriguing information about the criteria that certain rules are using. I ran Thor as the `chris` user, as opposed to the root user, and it still managed to dig very deeply into my laptop's filesystem. Most striking was a large chunk of alarming red text:

```
Alert Malware file found SCORE: 91
```

Clearly, this wasn't a welcome sign, however, I remembered that I was running Thor on the laptop that I use for Ethical Hacking, where the powerful Metasploit penetration testing framework [22] was installed. Upon closer inspection, I determined that the alarming output was relating to Metasploit payloads of varying descriptions.

The Metasploit payload that first showed up as a malware alert in Thor was a PowerShell version of our old friend `mimikatz` (Listing 4).

Listing 4: Thor Finds mimikatz

```
FILE: /opt/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/metasploit-payloads-2.0.148/data/meterpreter/
  ext_server_kiwi.x64.debug.dll EXT: .dll TYPE: EXE
SIZE: 1498112
[...snip?]
REASON_1: YARA rule Powerkatz_DLL_Generic / Detects Powerkatz - a Mimikatz version prepared to run in memory via
  Powershell (overlap with other Mimikatz versions is possible) SUBSCORE_1: 80 REF_1: PowerKatz Analysis SIGTYPE_1:
  internal SIGCLASS_1: YARA Rule MATCHED_1:
```

Listing 5: Thor Hits

```
REASON_1: Filename IOC \RottenPotato SUBSCORE_1: 85 REF_1: Relay Attack Tool Names SIGTYPE_1: internal SIGCLASS_1:
  Filename IOC MATCHED_1: /rottenpotato

REASON_1: Filename IOC \JuicyPotato SUBSCORE_1: 85 REF_1: Relay Attack Tool Names SIGTYPE_1: internal SIGCLASS_1:
  Filename IOC MATCHED_1: /juicypotato

REASON_1: Filename IOC \PetitPotam SUBSCORE_1: 90 REF_1: PetitPotam Names https://github.com/topotam/PetitPotam SIGTYPE_1:
  internal SIGCLASS_1: Filename IOC MATCHED_1: /petitpotam
```

Listing 6: Thor Report

```
MATCHED_1
    - WS2_32.dll at 0x48b66 in

"mIStorage\x00ole32.dll\x00\xa4\x00freeaddrinfo\x00\x00\xa5\x00getaddrinfo\x00WS2_32.dll\x00\x00)\
  x01EnterCriticalSection\x00\x00\xa5\x03LeaveCriticalSection\x00\x00"
    - ReflectiveLoader at 0x484d4 in

"@CAKPEAX@Z\x00?triggerDCOM@CMSFRottenPotato@@QEAAHXZ\x00ReflectiveLoader\x00\x00\x00\x00\xe0\x9c\x04\x00\x00\x00\
  x00\x00\x00\x00\x00\x00\x00\x08\x9e\x04\x00\x80S\x03\x00\x88\x99\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00@\x9e\x04\
  x00(P\x03\x00`\x99\x04\x00\x00\x00"
```

Examples of the other malware or IoC hits, which were mostly found either in Metasploit files or in backup files for the laptop, appear in Listing 5.

To keep things simple, Thor creates a helpful HTML file summarizing the analysis report (Figure 11).

As you can see from Figure 11, my laptop had a staggering 237 IoC, or malware, hits found by the excellent Thor.

The report contains a detailed information box for each alert that helps identify which match caused the alert, as shown in Listing 6.

To say that Thor (even the Lite version) is impressive, is an understatement. I plan on continuing my subscription for personal use and will recommend the full-fat version to any interested clients.

Defense by Thor

Thor can look beyond just signatures and covers all of the following security threats:

- Keyloggers
- Backdoors
- Remote Access Trojans
- Web shells
- Port scanners
- Hacking tools
- Anomalous system files
- Obfuscated scripts
- Proxy software
- Spyware
- Exploit codes
- Rootkits
- Credential stealers
- Privilege escalation tools
- Adversary activity
- Phishing attachments

Indicators of Compromise

Thor has taken the functionality provided by antivirus software and endpoint detection and response (EDR) tooling to another level. It also includes IoCs, which give security analysts useful pointers into suspicious activity. More importantly, the ability to fully automate such sophisticated scans is invaluable. And, don't forget that Thor lets you fully customize rules and create your own rules from scratch, allowing you to fit any edge cases in your environment that might not fit with public rulesets. See the box entitled "Defense by Thor" for more on some of the threats covered by Thor. The security coverage in Thor is significantly broader than traditional antivirus software and is well worth investigating further.

Conclusion

Hopefully, this relatively fleeting look at the excellent YARA has given you the impetus to try it out yourself. Many resources are available online, including an extensible set of online ruleset examples in public code repositories.

Having a malware-scanning tool that supports scripting is invaluable. I will leave you to explore the excellent Thor and inimitable YARA in greater detail and enjoy the security benefits as you go. ■■■

Author

Chris Binnie is a Cloud Native Security consultant and coauthor of the book *Cloud Native Security*: *https://www.amazon.co.uk/Cloud-Native-Security-Chris-Binnie/dp/1119782236*

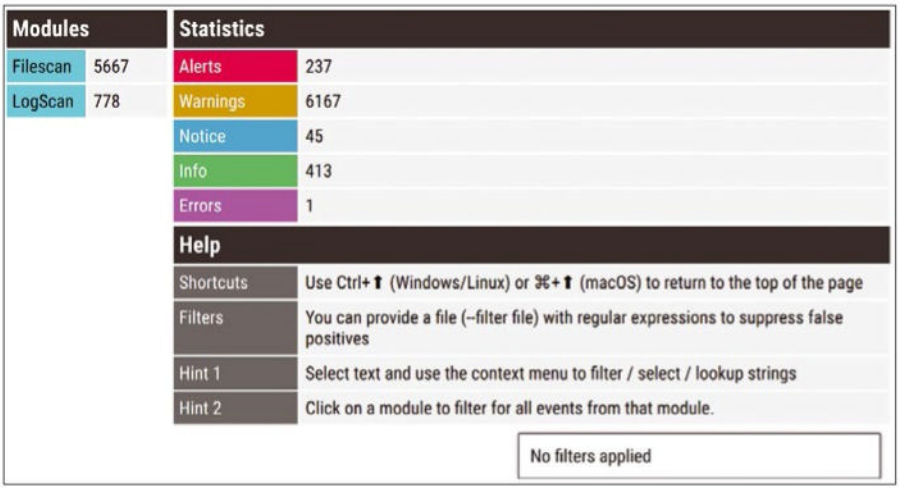


Figure 11: The summary section of Thor's excellent HTML report.

Info

[1] YARA: *https://github.com/VirusTotal/yara*

[2] VirusTotal: *https://www.virustotal.com*

[3] YARA license: *https://github.com/VirusTotal/yara/blob/master/COPYING*

[4] Google acquisition of VirusTotal: *https://techcrunch.com/2012/09/07/google-acquires-online-virus-malware-and-url-scanner-virustotal*

[5] YARA documentation: *https://yara.readthedocs.io/en/stable/index.html*

[6] YARA website: *https://virustotal.github.io/yara*

[7] YARA compiling docs: *https://yara.readthedocs.io/en/stable/gettingstarted.html#compiling-and-installing-yara*

[8] Mimikatz: *https://github.com/ParrotSec/mimikatz*

[9] Full Security Engineer post: *https://www.fullsecurityengineer.com/how-to-use-yara-to-detect-malware*

[10] YARA APT rule: *https://github.com/Yara-Rules/rules/blob/master/malware/APT_APT10.yar*

[11] Repos with YARA rules: *https://github.com/InQuest/awesome-yara*

[12] Operating YARA: *https://cocomelonc.github.io/tutorial/2022/02/15/malware-analysis-3.html*

[13] Commercial YARA tools: *https://www.virustotal.com/gui/home/upload*

[14] Infected Mimikatz executable: *https://github.com/ParrotSec/mimikatz/blob/master/Win32/mimikatz.exe*

[15] API reference: *https://developers.virustotal.com/reference/overview*

[16] Public vs. Premium API: *https://developers.virustotal.com/reference/public-vs-premium-api*

[17] Nextron Systems: *https://www.nextron-systems.com*

[18] Thor Lite: *https://www.nextron-systems.com/thor-lite*

[19] Loki: *https://github.com/Neo23x0/Loki*

[20] Premium signatures: *https://github.com/Neo23x0/signature-base*

[21] Valhalla ruleset: *https://www.nextron-systems.com/2018/12/21/yara-rule-sets-and-rule-feed*

[22] Metasploit: *https://www.metasploit.com*

MakerSpace

An AI module for the Pi 5

Turbocharge Your Raspberry Pi

What happens when the Raspberry Pi's makers and AI specialist Hailo collaborate on a project? We get an official AI kit HAT+ for the Pi 5 that adds an AI accelerator chip.

By Konstantin Agouros

The Raspberry Pi AI Kit [1] consists of two components: a generic M.2 HAT+, an adaptor board that lets you connect any two M.2 modules (e.g., for NVMe storage) to the Raspberry Pi 5 and therefore directly to the PCI bus, and a Hailo-8L AI accelerator, which could also be installed on any other computer with an M.2 interface. This entry-level AI chip achieves a performance of 13 TOPS, which is half the number of tera-operations per second that you'd get with the standard Hailo-8 model.

As a first step, you need to assemble the system. Fit the four spacers, a GPIO extension connector, and a PCIe extension cable on the Raspberry Pi. Then slot the M.2 HAT+ onto the spacers and plug the ribbon cable into the connector on the HAT. Figure 1 shows the fully assembled, working system.

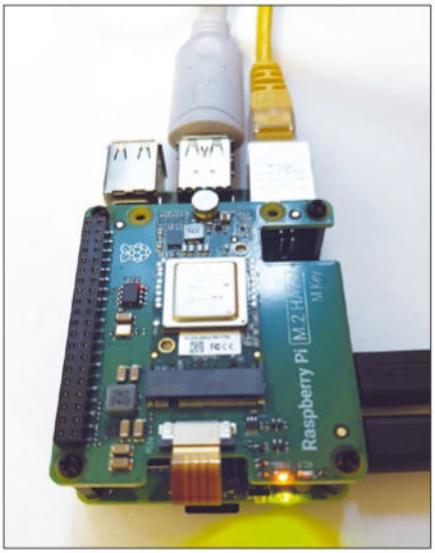


Figure 1: This is what the fully assembled AI HAT looks like.

After restarting the Raspberry Pi, the Hailo board appears in the `lspci` output as shown in Listing 1. The second line reveals that the installation of the AI accelerator was successful.

Listing 1: lspci Output

```
0000:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries BCM2712 PCIe Bridge (rev 21)
0000:01:00.0 Co-processor: Hailo Technologies Ltd. Hailo-8 AI Processor (rev 01)
0001:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries BCM2712 PCIe Bridge (rev 21)
0001:01:00.0 Ethernet controller: Raspberry Pi Ltd RP1 PCIe 2.0 South Bridge
```

What the Hailo Board Can Do

Hailo's AI accelerator modules are not generic accelerators for neural networks that could be used to reduce training times. Instead, they boost the operating speed (i.e., the reasoning and inference steps). It turns out that addressing the accelerator is not exactly easy.

Many machine learning (ML) applications use PyTorch or TensorFlow, and pre-trained models are usually tailored to one of the two frameworks. If you want to use either one with a hardware accelerator, you can call simple functions that load the neural network into the graphics card, using the CUDA interface if you have NVIDIA hardware. Then you can send input data to the graphics card and retrieve the response.

Using a Hailo board is different: You need a specially compiled and adapted version of the neural network. If you go to the Hailo website [2] and select *Products* | *Software*, you will find both pre-compiled networks (*Model Zoo*) and the *Dataflow Compiler*, which can convert common formats to Hailo Executable Format (HEF).

In the Hailo world, "building" does not simply mean compiling. Developers need to understand how to convert their neural networks into the desired format. Hailo boards, for example, can only handle integer values, so the network may need to be adapted. Detailed instructions and tutorials are available on the Hailo website. The target board also plays a role in the build process: You cannot load a Hailo-8 HEF file into the Hailo-8L and vice versa.

The main application for Hailo's AI accelerators is image recognition. According to the manufacturer, the boards can process significantly more frames per second than a Jetson Nano while using significantly less power at the same time. The performance of the Hailo-8 series boards is not sufficient for large language models (LLMs), but stepping up its game, Hailo has already announced the Hailo-10H M.2 Generative AI Acceleration Module, which is likely to be a useful choice for LLMs. Using a Raspberry Pi as a chatbot could be an interesting alternative, especially due to the Pi's comparatively low power consumption.

Software Setup

You have two options for installing the software. The simple approach, assuming you are using Raspberry Pi OS, is to install the *hailo-all* package with `apt`. The meta package from the Pi OS standard repository installs the *hailo-tappas-core-3.28.2*, *hailofw*, *hailort*, and *rpicam-apps-hailo-postprocess* packages. The software has reached version 4.19 in November 2024, but this release does not include Python modules to enable direct use of the Hailo board in Python. The manufacturer's code examples use GStreamer pipelines instead. You'll find the required plugins in the *hailort* package. After rebooting, run

```
hailortcli fw-control identify
```

The command checks whether the installation has succeeded. If the output looks like Listing 2, everything is OK. You can also use the `hailortcli` command to discover more of the board's details. For example, the `monitor` sub-command uses a `top`-like approach to continuously show you what the Hailo board is doing, which neural networks are running on it, and what level of data throughput it is achieving.

Listing 2: Function Check

```
$ hailortcli fw-control identify
Executing on device: 0000:01:00.0
Identifying board
Control Protocol Version: 2
Firmware Version: 4.17.0 (release,app,extended context
  switch buffer)
Logger Version: 0
Board Name: Hailo-8
Device Architecture: HAILO8L
Serial Number: HLDDLBB241901708
Part Number: HM21LB1C2LAE
Product Name: HAILO-8L AI ACC M.2 B+M KEY MODULE EXT TMP
```

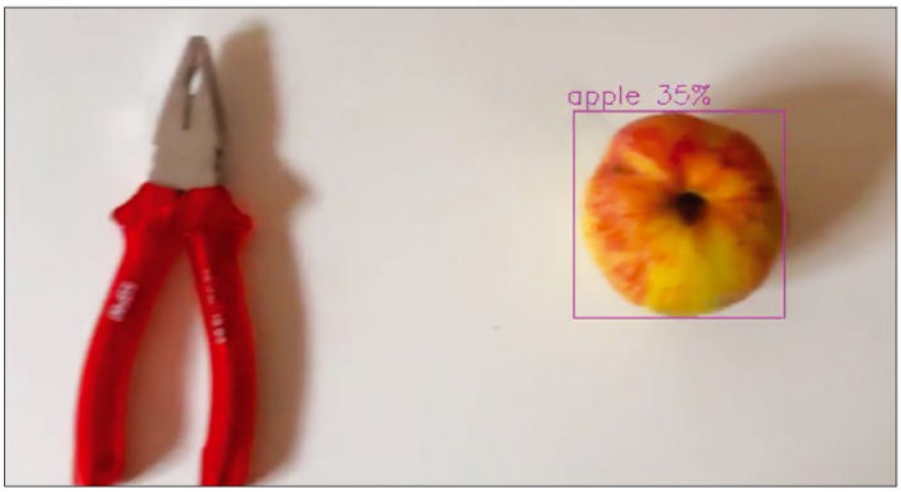


Figure 2: The apple in this image was recognized and properly labeled.

The second method of installing the required software involves a little more overhead. In Hailo's Developer Zone [3], you can find Debian packages and Python Wheel packages. (You need to create a Hailo account for accessing these files.) If you've previously installed the same version from some other packages, you first need to completely uninstall the existing packages. If you fail to do so, mismatching versions of the kernel module and the firmware will be installed and will then interfere with each other. Note that, in my own experience and that of some users in the Hailo community, the sample code described in the next section will no longer work if you take this route.

Examples

Start by running `git clone` to download the repository with the sample code [4] for the Raspberry Pi/Hailo combination. The `download_resources.sh` script that comes with the repository downloads the required models as `*.hef` files. The `setup_env.sh` script sets up a Python Virtual Environment inside which you can then install the required Python modules.

The `basic_pipelines/` folder contains three scripts that will give you some idea of what the Hailo board can do:

- `detection.py` is used for classic object detection in individual images or videos.
- `instance_segmentation.py` can be used to distinguish multiple objects in an image.
- `pose_estimation.py` extracts a "stick figure" or the 3D coordinates of several points of a person such as wrists, knees, feet, eyes, and nose from a video stream. The input can stem from a USB camera, a Pi Cam, or a file.

All three scripts build a GStreamer pipeline. It is called from Python and controls the Hailo hardware. The calling script specifies a Python function as a callback. The data determined by the network (such as the object name or the 3D coordinates) are passed to the callback which lets the program code manipulate the results.

The scripts give you some graphical output at the end of the pipeline, assuming that your Raspberry Pi is

running with a graphical user interface and either has a display or uses X forwarding to redirect the output. The output highlights objects detected by the network (Figure 2).

I have modified the code so that the script generates a video file (Listing 3). I'm sure that there are more elegant ways of doing this, but the results were good enough to see that recognition works as intended. If you used a null sink module that logs data without actually processing it, you could just run the callback and add code that triggers an action as soon as something specific is detected. Although this type of programming is a little roundabout, it can give you usable results.

You will find standard programming examples on Hailo's GitHub page [5]. This code runs with versions down to 4.18. Note that the download scripts provided there give you the HEF files in the format required for the Hailo-8, and not for the Hailo-8L. Also, there are no instructions on which model from the *Model Zoo* (which is also available on GitHub) you need to install for the scripts.

Listing 3: Code Edit

```
### Original code, commented out
# pipeline_string += f"fpsdisplaysink \
#   video-sink={self.video_sink} \
#   name=hailo_display sync={self.sync} \
#   text-overlay={self.options_menu.show_fps} \
#   signal-fps-measurements=true "
### Replacement code for video
pipeline_string += "matroskamux ! filesink \
  name=hailo.display location=video.mkv"
```

Conclusions

The Raspberry Pi AI Kit is relatively inexpensive and impresses with very low power consumption compared to a graphics card. The boards are only intended for inference (i.e., they cannot help you to train your networks during ML development). This limits their potential application scope. Also, getting a trained neural network to run on the board isn't exactly easy.

At the end of the day, though, the AI Kit is a great choice for image recognition applications – thanks to its combination of performance and low power consumption – and it perfectly complements the Raspberry Pi.

There is an interesting announcement on the Hailo website relating to the Hailo-10H board. The manufacturer claims that this board is suitable for running LLMs in the style of ChatGPT (but admittedly only its smaller siblings). If this turns out to be true, you could run an LLM on a Raspberry Pi, which would be a very interesting option due to the Pi's low power consumption. ■■■

Info

[1] Raspberry Pi AI Kit: *https://www.raspberrypi.com/documentation/accessories/ai-kit.html*

[2] Hailo: *https://hailo.ai/*

[3] Hailo software downloads: *https://hailo.ai/developer-zone/software-downloads*

[4] Hailo's Raspberry Pi examples: *https://github.com/hailo-ai/hailo-rpi5-examples*

[5] General examples for Hailo boards: *https://github.com/hailo-ai/Hailo-Application-Code-Examples*

Author

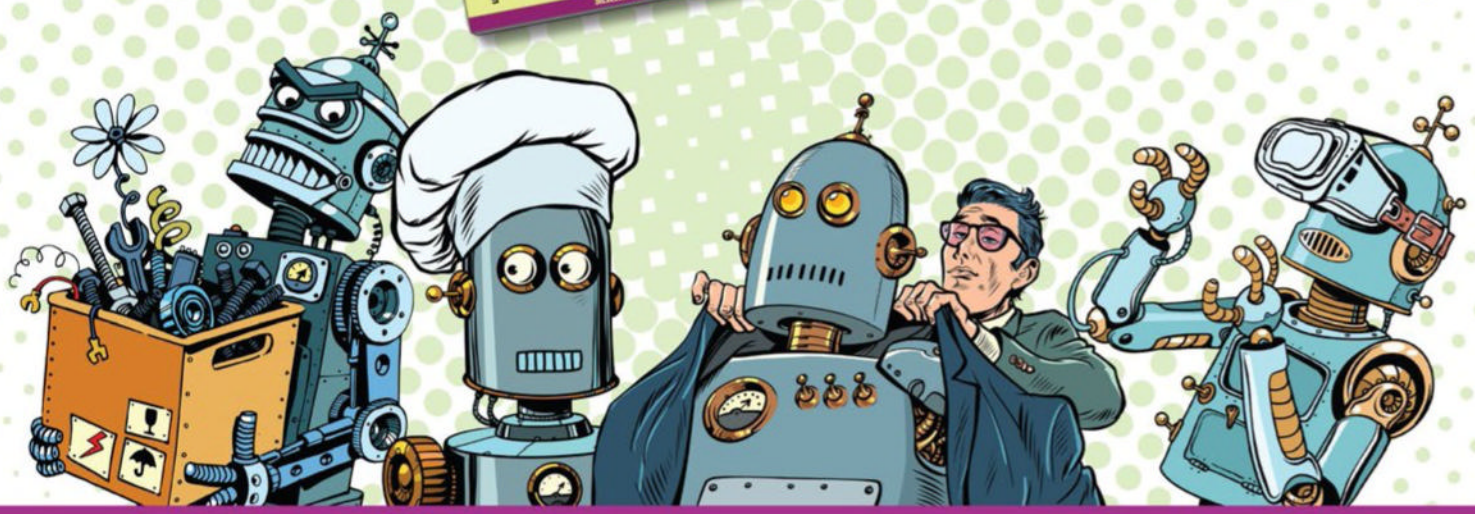
Konstantin Agouros works as Head of Engineering & Security Solutions at X1F GmbH. Together with his team, he advises customers on topics relating to open source, security, and the cloud. He is the author of a (German) book on Software Defined Networking that was published by De Gruyter.

Turn your ideas into reality!

This is not your ordinary computer magazine! *MakerSpace* focuses on technology you can use to build your own stuff.

If you're interested in electronics but haven't had the time or the skills (yet), studying these maker projects might be the final kick to get you started.

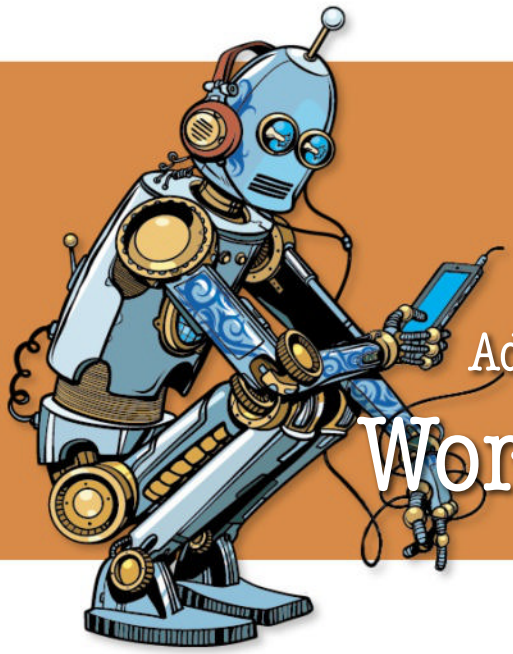
Issues available now!



Lead Image © Valeriy Kachaev and studiostoks, 123RF.com

COLLECT ALL 4

ORDER ONLINE: shop.linuxnewmedia.com



MakerSpace

Add animated drawings to your IoT dashboards

Worth a Thousand Words

Use an SVG graphic widget on your next Node-RED project to get a visual representation of your automation setup.

By Pete Metcalfe

There are some great components for showing tables, gauges, and charts on Internet of Things (IoT) dashboards. For many applications these data widgets are all that you need – however, for industrial or process monitoring projects, it can be extremely helpful to add dynamic visual layouts of the equipment. Figure 1 is an example of a process automation screen that could be used for an industrial control system.

Node-RED [1] is a powerful open source graphical programming tool that you can use for creating IoT logic scenarios and user dashboards. It runs on Linux, macOS, and Windows, and it's an excellent tool for prototyping or building small system solutions. Typical Node-RED projects use the standard dashboard components, but it's also possible to install an SVG widget that lets you show animated graphics alongside the other dashboard objects. Scalable Vector Graphics (SVG) [2] is an XML-based vector image standard that is supported on all major web browsers. SVG files support dynamically changing text, color, rotation, and scaling of objects.

In this article, I'll show you how to use Node-RED with the SVG widget to animate a small Raspberry-Pi-based home watering system.

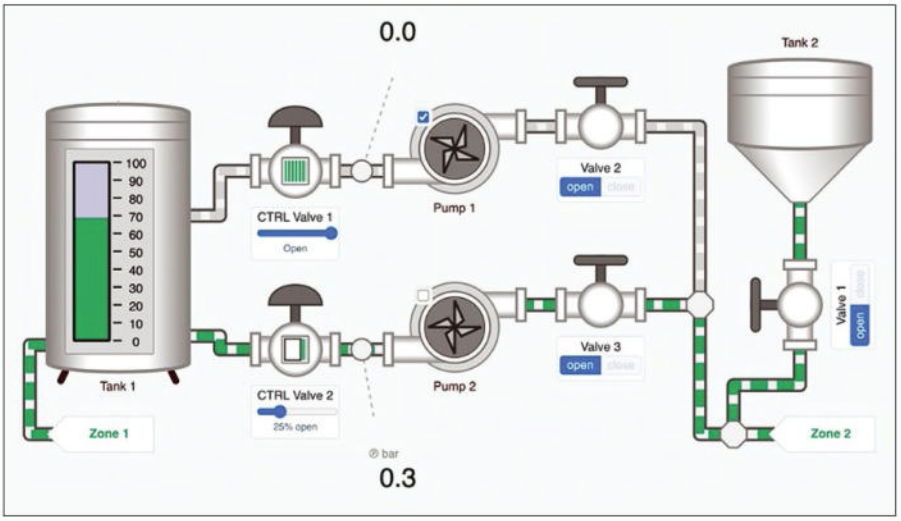


Figure 1: This is a typical process automation graphic with valves and pumps.

Getting Started

In the *Get Started* section of the Node-RED homepage [3], you can find specific installation directions for your hardware. You should use the latest version of the Node-RED web dashboards – check the version in the *Manage Palette* option. Add the SVG graphic component (*Node-RED-contrib-ui-svg*) via *Manage Palette* and select the *Install* tab. Once you've installed the component, you can drag and drop it on the *Flow* page like any other widget. The SVG flow has a built-in graphic editor, and you can also insert code manually in the *SVG* tab (Figure 2). The component also supports animations (for

simulation), event handlers, JavaScript integration (from within the SVG drawing), and binding links between SVG items.

SVG Pump Example

As a first example, I wanted to build a basic on/off pump. When creating simple graphics, I found that I had more control writing the code manually rather than using the Node-RED SVG graphic editor. You can write SVG files in any text editor and view the resulting graphic presentation in any browser.

SVG files start with an `<svg>` tag and end with an `</svg>` tag (Figure 3). The opening tag also needs to have `xmlns` (namespace) and `xmlns:xlink` parameters (lines 3 and 4). The pump example consists of four elements: two horizontal rectangles for the in and out pipes (lines 9-10), a large circle for the pump body (line 12) and a smaller circle for the inner motor shaft (line 13). A group tag (`<g>...</g>`) combines all elements, and its `id` parameter gives the group a logical name so that it can be referenced from other code. In this example, the pump group's `id` name is `body`.

It is important to note that no colors were defined for any of the shapes, except the inner circle (pump motor shaft), because the inner circle's color will remain unchanged while the rest of the pump body will be set dynamically in the next step.

My hardware setup uses a Raspberry Pi with a Pimoroni Automation Hat, but any relay module would work (Figure 4). A relay offers isolation between the high power requirements of the pump and the Raspberry Pi. After connecting the hardware to the Raspberry Pi, I had to install the Node-RED GPIO widget. Typically the GPIO components are included in the Raspberry Pi Node-RED installation – if not, you can add them manually via:

```
cd ~/.Node-RED
npm i Node-RED-node-pi-gpio
```

The Node-RED logic for this pump project uses only five nodes to show the SVG graphic and control the pump (Figure 5): I use two *dashboard* buttons, *RUN* and *STOP*, for user input to send a `1` or `0` to the *GPIO* node, on pin 13. The *SVG graphics* node had the manually created pump code pasted into the *SVG* tab.

You can modify SVG elements by passing commands in the message payload. Some of these commands are `add_event`, `set_attribute`, `pan`, `trigger_animation`, `update_style`, `update_text`, and `zoom`. See the SVG graphic node documentation for more details.

The *Function* node (Listing 1) reads the user input (`msg.payload`, line 5) to define a new pump color. It then outputs a new `msg.payload` that includes the `update_style` command, which changes the fill color of the SVG pump selected by the `#body` id (lines 8-13).

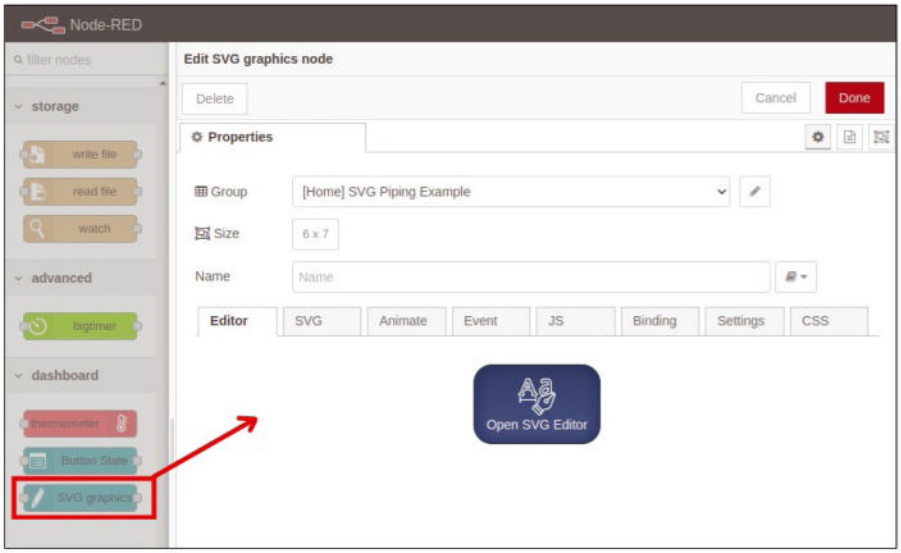


Figure 2: The SVG component has an editor, and you can also import SVG files.

```
1  <!-- SVG Pump in a 300x300 window-->
2  <svg width="300" height="300"
3    xmlns="http://www.w3.org/2000/svg"
4    xmlns:xlink="http://www.w3.org/1999/xlink">
5
6    <!-- pump body with in/out line-->
7    <g id="body" fill="black">
8        <!-- in/out line-->
9        <rect width="140" height="40" x="10" y="130" />
10       <rect width="140" height="40" x="150" y="30" />
11       <!-- pump body with inner shaft in silver-->
12       <circle cx="150" cy="100" r="70" />
13       <circle cx="150" cy="100" r="30" fill="silver"/>
14   </g>
15
16 </svg>
```

Figure 3: These SVG lines define the simple pump shown on the right.

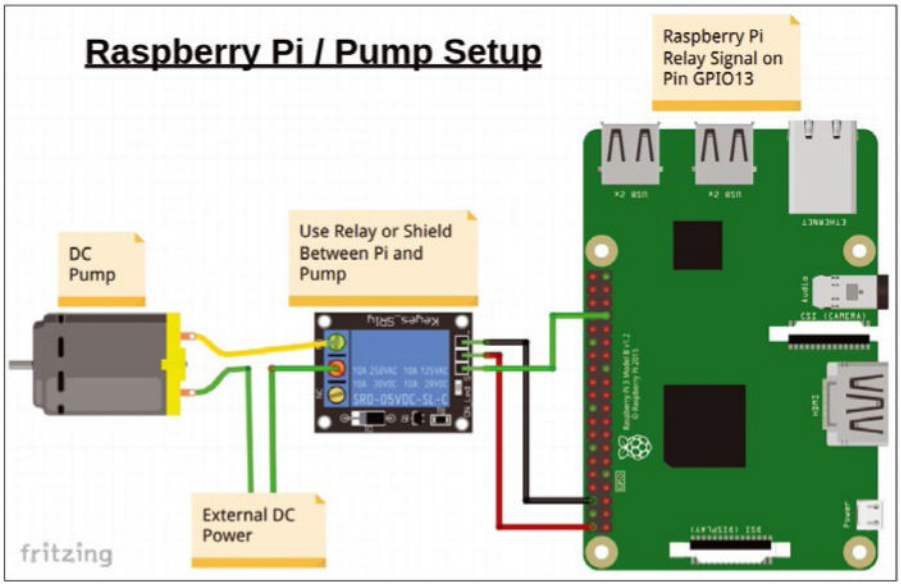


Figure 4: The Raspberry Pi uses a relay module to control a pump.

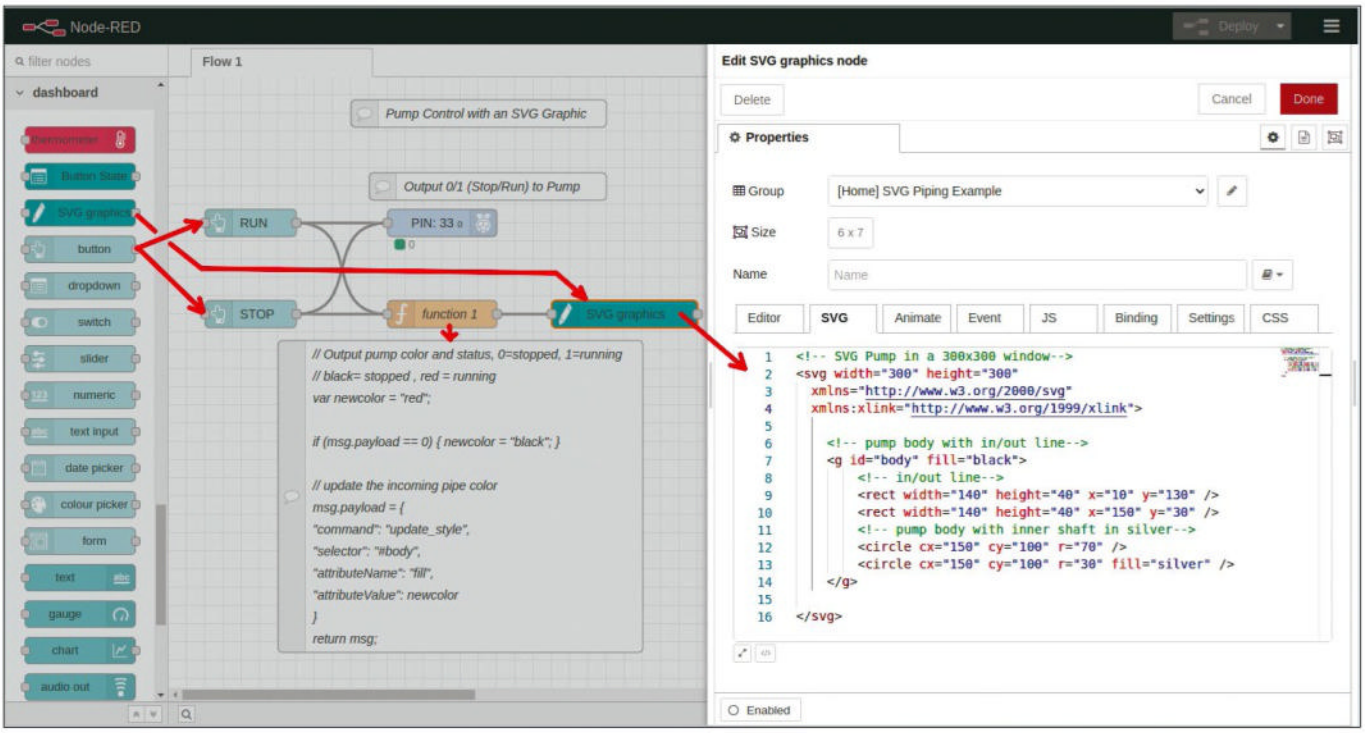


Figure 5: The custom SVG file for the pump leaves the color undefined; it can later be set via a function.

Listing 1: Update SVG Pump Color

```
01 // Output pump color and status, 0=stopped, 1=running
02 // black= stopped , red = running
03 var newcolor = "red";
04
05 if (msg.payload == 0) { newcolor = "black"; }
06
07 // update the incoming pipe color
08 msg.payload = {
09     "command": "update_style",
10     "selector": "#body",
11     "attributeName": "fill",
12     "attributeValue": newcolor
13 };
14 return msg;
```

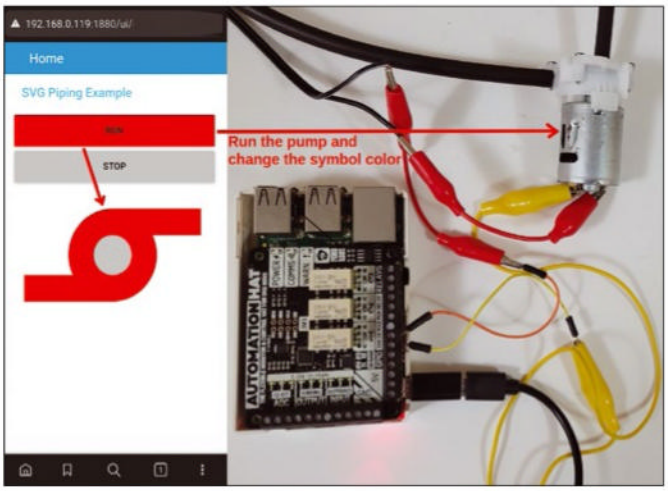


Figure 6: This Node-RED interface turns the water pump (on the right) on and off.

Figure 6 shows the running Node-RED pump dashboard with the Raspberry Pi test setup. When you select the *RUN* button, the relay is energized (state = 1) and the pump starts; in addition, the SVG pump color changes from black to red (the energized state). As a single pump is not overly useful, I'll add some valves and piping in the next example.

Valve Symbols

In the pump graphic, the pump is a grouped entity. This approach works well for single use objects, but creating a symbol is a better solution for projects that require multiple instances of an item. The SVG element `<defs>` is used to store graphical objects that will be used at a later time. Inside the `<defs>...</defs>` tags, I can define symbols.

In the next example I define a valve symbol (Figure 7, lines 7-10): The shape of the valve is a `<polyline>` that crisscrosses to create two triangles (line 9). The `<use>` element references

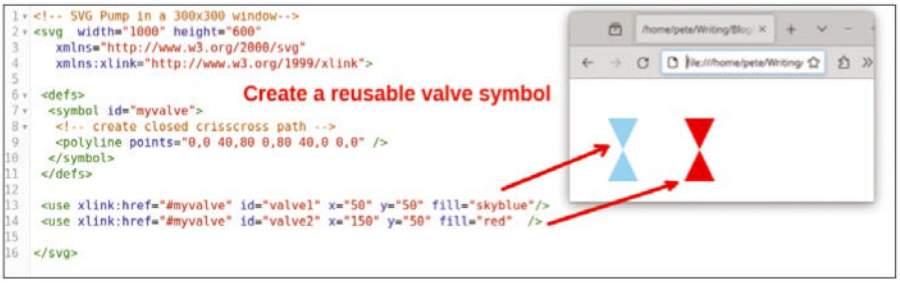


Figure 7: The SVG valve symbol `myvalve` is reusable via a `<use>` element.

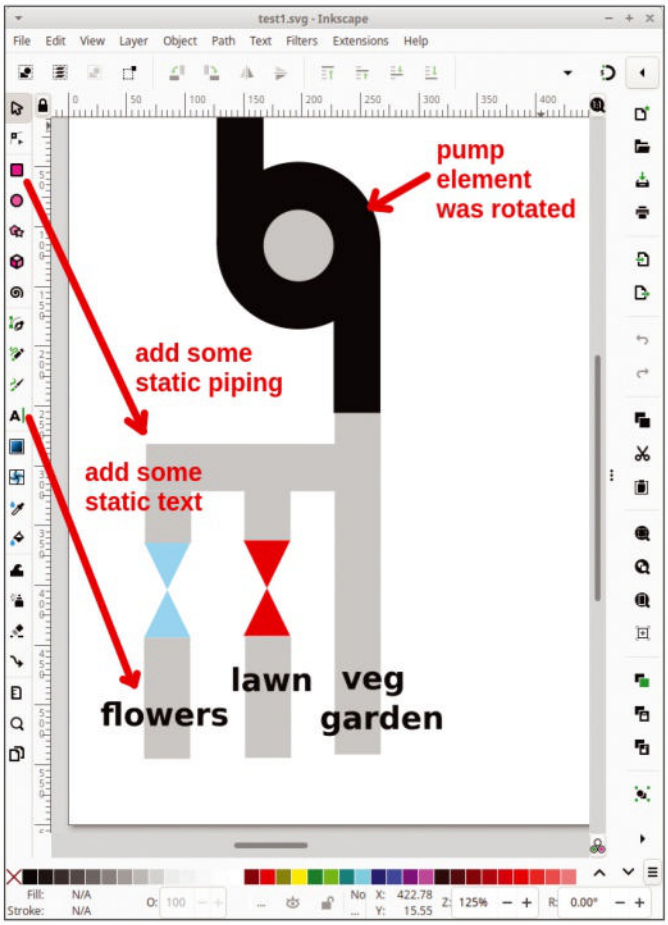


Figure 8: Inkscape can merge and modify several SVG files. Here, I have rotated the pump by 90 degrees and created some static piping.

symbols, and it sets their x,y coordinates while also passing scaling, rotation, and configuration parameters like fill color (lines 13-14). It's important to give each symbol instance a new ID name (e.g., `valve1`, `valve2`, etc.).

After testing the valve symbols, the next step is to merge the pump and the valves files and then add some pipes to connect things. Unfortunately, the Node-RED SVG editor isn't quite designed for this, but luckily there are a number of free apps like Inkscape [4] that can do this (Figure 8).

For the Node-RED pump/valve example, I simplified the configuration so that I can focus on the SVG animation. To keep things lean, I used a *Button State* widget that creates a single object which is an array of buttons instead of working with six individual buttons. You need to install the widget with:

```
npm i Node-RED-contrib-ui-button_state
```

The second example requires only three nodes (Figure 9): The *Button State* node contains the user interface for turning the pump on and off and the open/closed states for the two valves. The *Function* node, as in the first example, changes the fill color of SVG elements. Finally the *SVG graphics* node is updated with the new SVG code that I created in Inkscape.

I've set six buttons for the *Button State* node (Figure 10). The *PUMP ON* and *VALVE n OPEN* get red backgrounds, while the *PUMP OFF* and *VALVE n CLOSE* buttons use silver. The *Button State* node's `msg.payload` will send a string with the requested value and the ID name. As an example, pressing *PUMP ON* will pass a "`1body`" payload, and *PUMP OFF* outputs "`0body`".

The *Function* node will parse the *Button State* node's `msg.payload` to find the requested state and ID name and then update the SVG graphic accordingly (Listing 2). Lines 5-6 use the JavaScript `slice()` function to get the first character and the rest of the string. Line 14 inserts the ID name of the passed SVG element (`thename`) into `msg.payload`.

The final dashboard (Figure 11) lets me toggle the states of the pump and the two valves. A refined version includes a useful background image that shows the real-world objects affected by the control mechanism.

Listing 2: Six Button Array

```
01 // Slice the msg.payload
02 //   first character is 0 or 1,
03 //   the rest of the string is the ID name to modify
04
05 var thevalue = msg.payload.slice(0,1);
06 var thename = msg.payload.slice(1,);
07
08 var newcolor = "red";
09 if (thevalue == 0) { newcolor = "black"; }
10
11 // update the requested SVG element color
12 msg.payload = {
13     "command": "update_style",
14     "selector": "#" + thename,
15     "attributeName": "fill",
16     "attributeValue": newcolor
17 };
```

More Complex Graphics

The examples I've shown so far use extremely simple SVG graphics. Creating complex drawings from scratch can be a

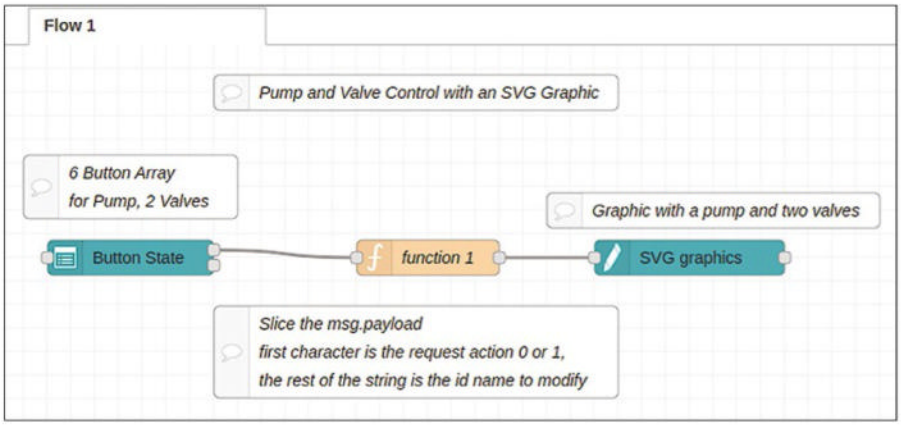


Figure 9: This logic controls a pump and two valves.

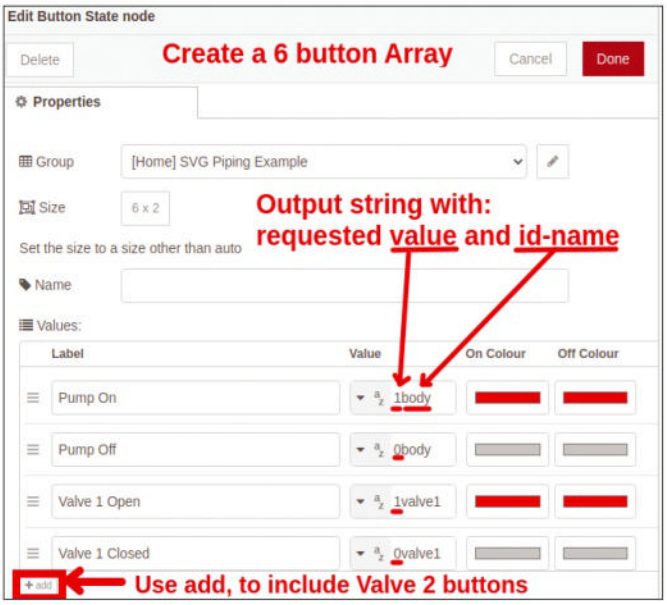


Figure 10: A dialog helps with defining a six button state array.

challenging and time-consuming exercise. Luckily, you can download files from free SVG libraries that can greatly speed up your design efforts. Some industrial process control vendors (e.g., Opto 22 [5]) offer high resolution graphics for free or a low price. You can merge downloaded SVG files with your existing files in the same way as earlier with the pump and valves files. Use apps like Inkscape to resize, rotate, or reposition objects.

One of the challenges in working with downloaded SVG files is identifying the pieces and parts that you want to manipulate in your application. Figure 12 shows how the built-in SVG graphic editor is used to isolate the front panel on an industrial cooling unit. Once the required element is identified, it can be given an ID name and then accessed from code.

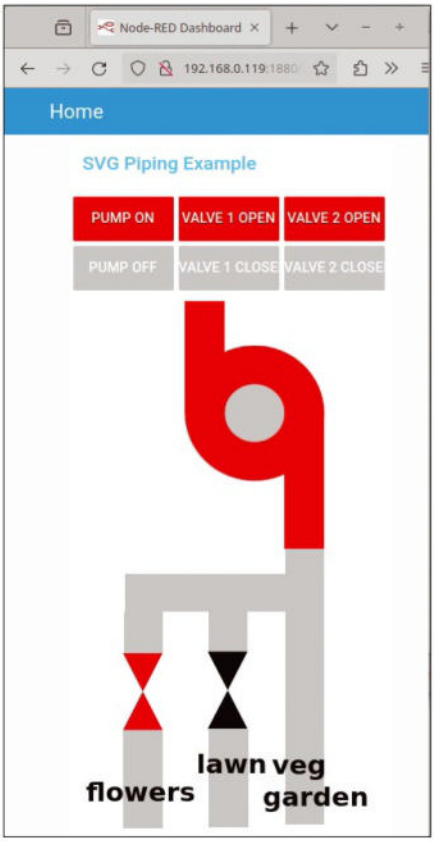


Figure 11: This dashboard controls a pump and two valves via the six buttons.

Summary

I've briefly covered creating basic symbols. Understanding the SVG syntax will greatly improve your efforts in designing graphics. For example, if you can download a free SVG file, you can add a `<symbol>` tag around a selected element and then reuse the image multiple times in your overall drawing. The Node-RED SVG widget has some incredibly powerful features that are worth investigating further – I've only discussed the `update_style` command, but there are many others.

A good use case for Node-RED SVG graphics might be to overlay your house floor plan with equipment such as lighting, temperature, cameras, and other powered devices. ■■■

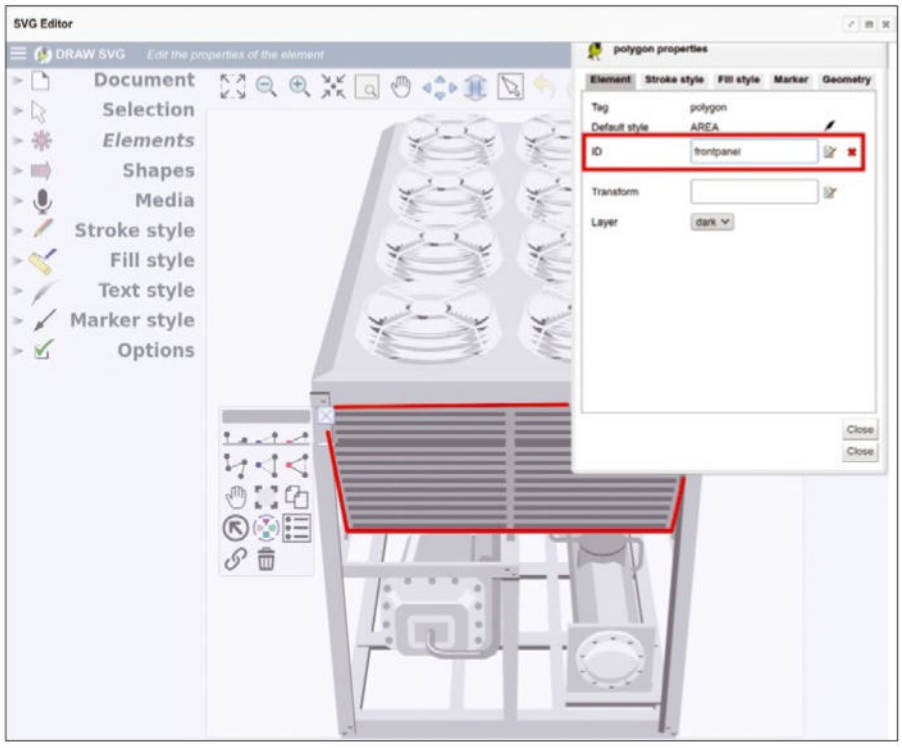


Figure 12: Use the SVG editor to find and name objects in complex drawings.

Info

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

[2] SVG: *https://en.wikipedia.org/wiki/SVG*

[3] Node-RED, Getting Started: *https://nodered.org/#get-started*

[4] Inkscape: *https://inkscape.org/*

[5] Opto 22 SVG graphics library: *https://www.opto22.com/support/resources-tools/demos/svg-image-library*

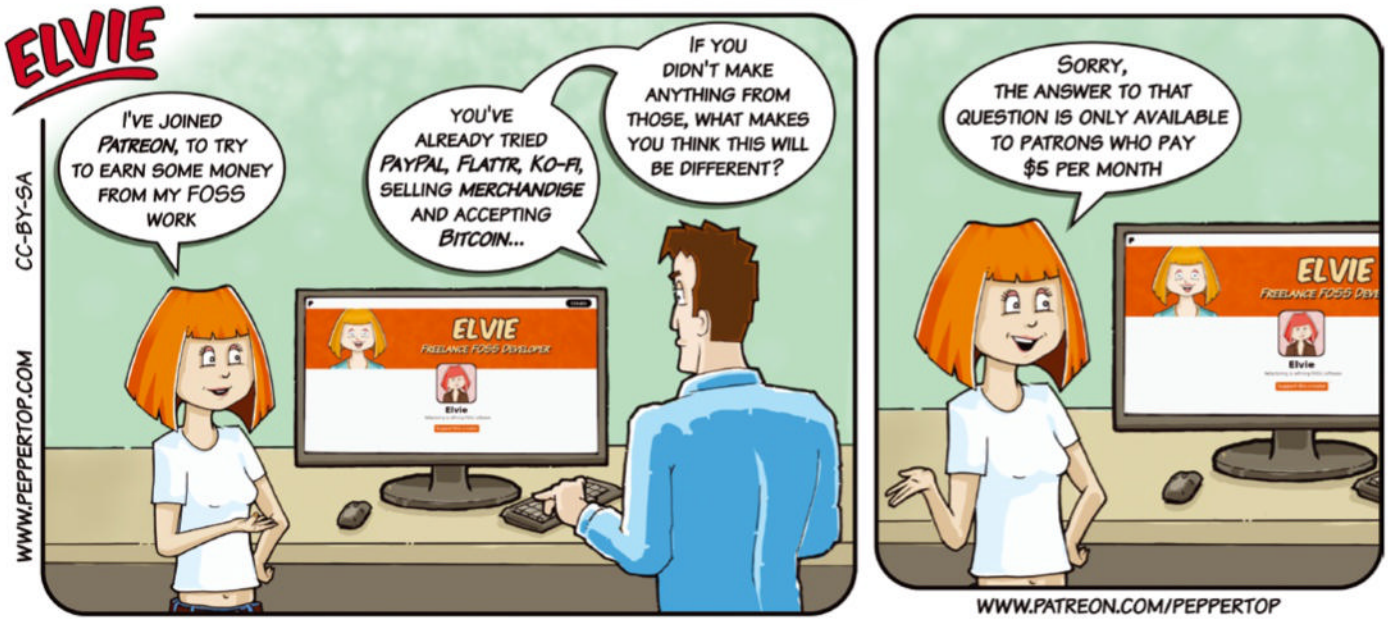
Author

For more of Pete's project, see: *https://funprojects.blog*.

Somewhere down inside, Android is just another Linux, right? And Linux can talk to Linux. Can my smartphone and my Linux desktop make friends? You can imagine the scenarios where it would be useful to control your Android from a Linux computer. Troubleshooting? Screen sharing? File sharing? As with anything else in open source, if there is a need, there is probably a tool. In this month's Linux Voice, we examine some applications that help you connect your Linux computer with an Android device. Also inside, we show you how to transfer data on a local network with LANDrop, and we introduce you to Piwigo, a tool for creating and organizing online photo galleries.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



MADDOG'S DOGHOUSE

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®.

Using your time to introduce someone to Linux can enrich their world and the free software community – and improve your own understanding in the process. BY JON "MADDOG" HALL

The gift of free software

This column was inspired during the Christmas season. That season will have passed when this column reaches you, but the sentiment of gift giving lasts the whole year through. Over the course of 2024, there was a lot of focus given to income differences, with some people having a lot of money and some people having very little.

One gift that can be given that is the same for both the mega rich and the not-so-wealthy is the gift of your time. More or less, people are granted a fixed length of time on this planet, and when that time is up we die. So every hour is a gift to give to someone which cannot be given again no matter how much money you have.

One of the things that keeps me going are the people who have come to me from time to time and thanked me for the time I have spent with them, either individually or in groups, to help them get started with free software. Sometimes I did not even realize the influence I was having on them, but often they turned my direction into positive actions in their own life.

What should this mean for you? If you are reading these words, you probably are a Linux user. Perhaps you have been using Linux for a year or two years, five years, or even more. I have been using Linux since May of 1994 and Unix since 1980. While I know a fair amount about Linux, I know there are people that know much more than I do, particularly about various specific parts of Linux. I admit to "holding on by my fingernails" when it comes to networking, because when I started out in computing "networking" was carrying a box of punch cards down the hallway or having a beer with friends at a conference.

However, I love taking a person who knows nothing about Linux and getting them started down a path of a lifetime of learning, particularly if it is someone I know from some other situation – a family member or a friend. I enjoy spending time with them.

The other thing about teaching someone free software is that it helps cement concepts in your own mind. If you study something on your own you may not understand a particular point, but you say to yourself, "That is OK, I will come back and understand this in the future." It requires a focused and driven student who actually does return to learn that point. However, when you realize you are going to be teaching this subject to someone else, you know that you have to understand that point because that is the one thing they will ask you about, and you will feel you have to know it.

On the other hand, one thing you can also teach someone is how to find the information they need by themselves: How to formulate a good Internet search, how to dig down and find the best answer to their question. This has the added value of not just "giving them the fish" but "teaching them how to fish."

One of the issues of closed source software is that the users will never be as knowledgeable as the developers who wrote it. I tell people new to Linux that because all of the source code is there, they too can read the kernel code and someday they could be almost as good a coder – or perhaps even better – as Linus Torvalds. John Lions, a former professor at the University of New South Wales, thought the best way of teaching students how to write good code was to show them the code of good coders. For a time, intellectual property laws stopped John from publishing what was probably one of the most valued technical books of that age, but it was eventually published and is still very valuable.

Teaching someone about free software does not need to be limited to Linux. There are lots of other pieces of free software that are databases, graphics programs, audio interfaces, and so on that you can learn and then learn even better by explaining them to someone else.

That person can then learn the software even better by showing and explaining it to other people, and so the free software community grows, one gift of time after another. ■■■

Controlling your Android device from the Linux desktop

Mind Meld

You don't have to be an expert to imagine situations where it would be useful to control an Android phone or tablet from the Linux desktop. Whether you are syncing notifications, sharing files, or even mirroring the screen, Linux has the tools for the task. BY ALI IMRAN NAGORI

Controlling your Android from Linux opens up exciting possibilities. Perhaps you want to browse the Android directories for a file? Maybe you wish to view a video captured on the Android? Linux provides several useful tools for connecting Linux with an Android device. This article rounds up some of the favorites.

In order to undertake these exercises, you'll need:

- a Linux box (I'm using Ubuntu 24.04, but any modern distro will do)
- an Android device (I'm using my phone)
- a USB cable
- a fair Internet connection

Android Debug Bridge (ADB)

The Android Debug Bridge (ADB) connects your computer to an Android device for advanced access and functionality [1]. The debug bridge is a command-line tool you install on your Linux system that lets you install apps, move files, debug issues, and execute other tasks on your Android phone. Install ADB on a Debian-based system as follows:

```
$ sudo apt update && sudo ➥
  apt install android-tools-adb ➥
  android-tools-fastboot<C>
```

For the ADB to be able to discover your device, you need to set up USB debugging on your Android system:

1. Head to *Settings*.
2. Tap *About Phone/Tablet*.
3. Find the build number and tap it seven times.
4. When you see a message that says *You're now a developer!* go back to *Settings*.
5. In *Settings*, search for *Developer Options* and toggle *USB Debugging* to *On*.

After you have installed ABD and configured your Android for debugging, connect the Android device to your computer using a USB cable. Once you've connected your device, you can use various ADB commands from your Linux system to control the Android phone. To list the devices available for debugging, enter the following command:

```
$ adb devices
```

You might get an error stating that you have insufficient permissions (Listing 1). If you get this message, you'll need to set up the Linux system so that it has permission to access the Android device through USB. Go to the Linux terminal and use the `lsusb` [2] command to see the devices connected to USB buses (Figure 1).

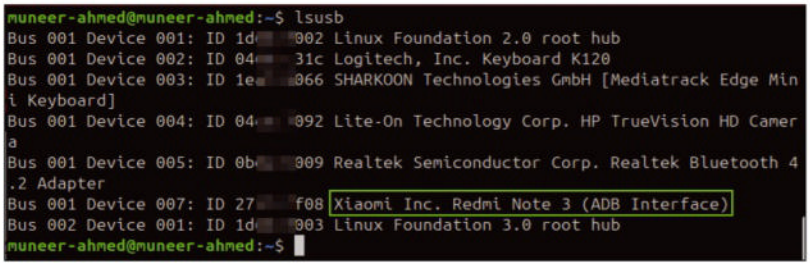
Locate your Android device and note the vendor ID and product ID. In my case, you can see a Xiaomi phone. The hexadecimal pair after the ID label is what I'm looking for. The first part is the vendor ID and the second one is the device ID or the product ID.

Now you just need to add a Udev rule for your Android device so the Linux system can access the Android phone. Udev [3] is a part of Linux that detects devices you connect to with your

Listing 1: Listing Devices

```
$ adb devices
List of devices attached
b177f7b70323    no permissions (missing udev rules?  user is in the plugdev
                group);
 see [http://developer.android.com/tools/device.html]
```

Figure 1: Listing USB devices with the `lsusb` command.



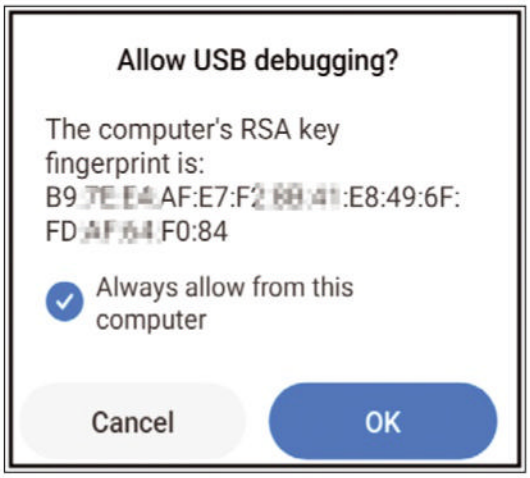


Figure 2: USB debugging pop-up message on an Android device.

computer. Open up a terminal and add the rule shown in Listing 2 to the `/etc/udev/rules.d/51-android.rules` file.

Finally, reload the rules to activate the new rule for the Android device:

```
$ sudo udevadm control --reload-rules
```

Now see if ADB can see your device:

```
$adb devices
```

If you still see something odd, unplug/replug the device. Also, try restarting the ADB server:

```
$ sudo adb kill-server
$ sudo adb start-server
```

If everything goes well, you'll see an *Allow USB debugging?* notification on your phone. Just accept that fancy pop-up message, and you're ready to move on (Figure 2).

Now again list the ADB devices and you should see the unauthorized notice is gone now:

```
$ adb devices
List of devices attached
c26***75        device
```

Here's where things get really interesting. I'll give some examples of commands you can use to control your Android system from your Linux computer.

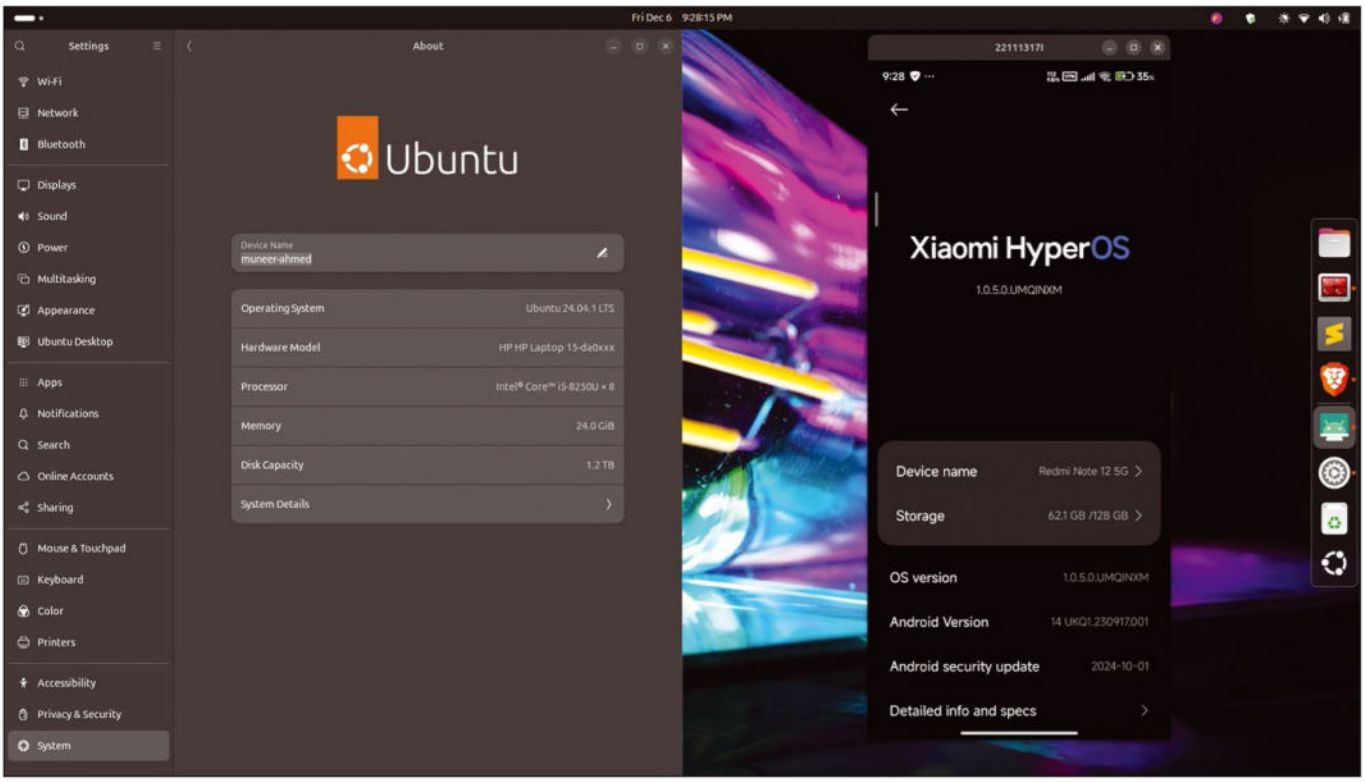
To take a screenshot of what's on your device screen, use:

```
$ adb shell screencap -p /sdcard/screenshot.png
$ adb pull /sdcard/screenshot.png
```

Listing 2: Adding a Udev Rule

```
$ sudo nano /etc/udev/rules.d/51-android.rules
SUBSYSTEM=="usb", ATTR{idVendor}=="[Enter_Vendor_ID]", ATTR{idProduct}=="[Enter_Product_ID]", MODE="0666", GROUP="plugdev"<C>
```

Figure 3: My Android phone mirrored on Ubuntu desktop.



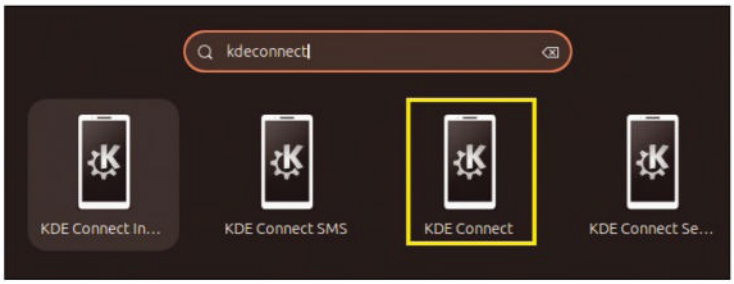


Figure 4: KDE Connect Linux app.

The preceding commands are self-explanatory; the first one takes a screenshot of the current window and saves it to the SD card. With the second command, the screenshot will be saved right to the Linux machine.

Screen recording is also pretty easy:

```
$ adb shell screenrecord /sdcard/video.mp4
```

Again, to get the recording on your Linux system, use the `pull` subcommand:

```
$ adb pull /sdcard/video.mp4
```

Do a reboot to refresh your device when it feels sluggish:

```
$ adb reboot
```

You can even experience a Linux-like environment on your phone. Start a shell session with

```
$ adb shell
```

From the shell interface, you can run many Linux-like commands. OK, I'll stop here. You'll find many more ADB commands to tinker with, but I think these are enough to grasp the potential.

A word of caution: ADB is powerful, but with great power comes great responsibility. Misusing commands like ADB shell or uninstalling critical apps can leave your device unstable – or worse. Always proceed with caution and know what you're doing.

Scrcpy: A Screen Mirroring Tool

Although ADB is a powerful tool, it can be a bit complex for everyday use. For a more user-friendly experience, you can use Scrcpy [4]. This tool lets you mirror and control your Android device from your Linux desktop, and it's ridiculously efficient.

You can install Scrcpy using the Ubuntu package manager; however, as of this writing, the Ubuntu version is obsolete. Get the latest version from the GitHub repo. First, download the release and extract it:

```
$ wget https://github.com/Genymobile/scrcpy/releases/download/v3.0.2/scrcpy-linux-x86_64-v3.0.2.tar.gz
$ tar -xzf scrcpy-linux-x86_64-v3.0.2.tar.gz
```

You'll need to repeat the steps that involve setting up USB debugging, which I discussed in the ADB case earlier. Additionally, you might need to enable USB debugging (*Security Settings*) for some phones like Xiaomi to control the device using a keyboard and mouse. Follow the reconfiguration by rebooting your Android device.

Now, head to the extracted directory and look for the *scrcpy* binary, and run it:

```
$ ./scrcpy
```

Accept any pop-up on your Android device that requests authorization. This will start mirroring your device's screen to your computer (Figure 3). You can then control the device using your mouse and Keyboard.

As you can see, Scrcpy just cloned your phone on your desktop.

KDE Connect: Simple, Yet Elegant

If you don't need a full-fledged feature-packed tool, KDE Connect [5] is the answer. KDE Connect can give you basic control of your Android device, allowing you to see notifications, control music, transfer files, and simple stuff like that. Don't let the KDE name fool you; it works great

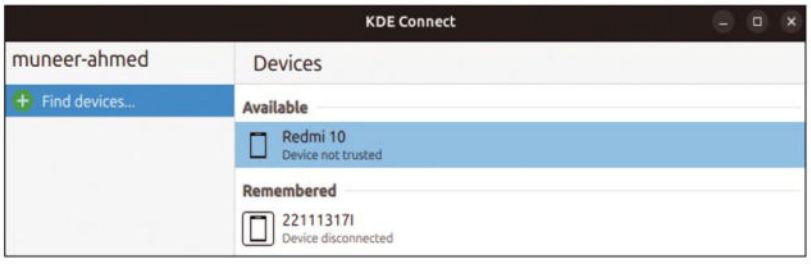


Figure 5: KDE Connect showing available devices.

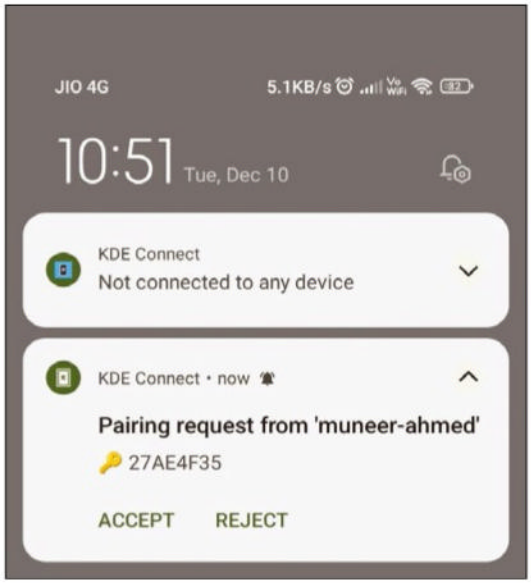


Figure 6: Pairing request on the Android device.

Figure 7: Options for controlling an Android device from the desktop.

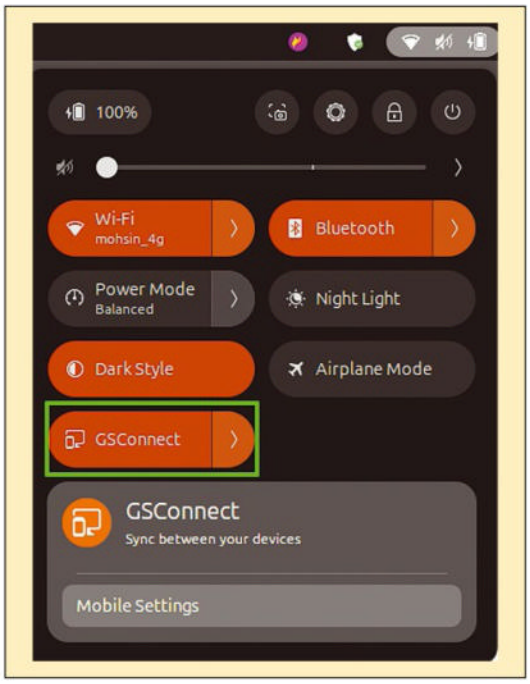


Figure 8: GSConnect in Gnome top bar settings.

Figure 9: GSConnect showing the device ID (unpaired for now).

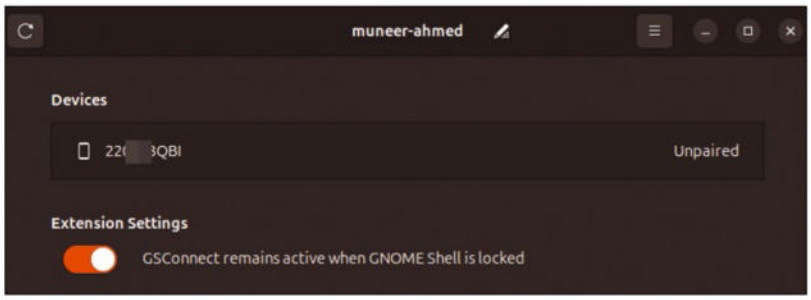
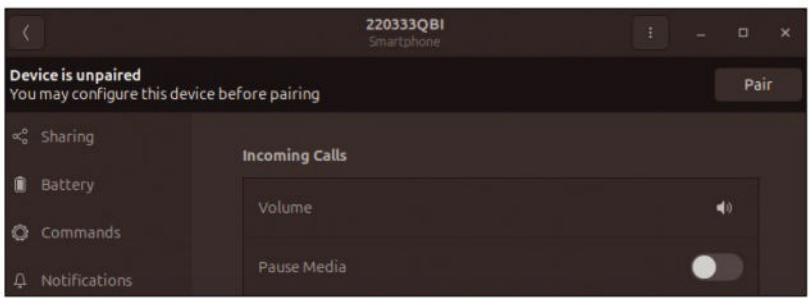


Figure 10: Initiating pairing from the desktop.



on Gnome as well. However, it best fits with a KDE desktop. To install KDE Connect:

```
$ sudo apt install kdeconnect
```

Now you can start it right from the *Application* menu (Figure 4). Install the KDE Connect app on your Android device from the Play Store. After installing the app, you'll see the name of your Linux desktop on the app when you start it (assuming both devices are on the same network). In the same way, the name of your Android device appears on the desktop app. In my case, it is a Redmi 10 phone (Figure 5).

Now pair the devices using the pairing button inside the device name. Next, confirm the pairing request from your Android device (Figure 6).

If the pairing is successful, you'll get various options on your PC to control your device (Figure 7). You can try some operations to check the potential of KDE Connect:

- send files between devices
- find your phone by ringing it
- use your device as a remote control for controlling multimedia
- share clipboards between devices

GSConnect: A Gnome Shell Alternative

GSConnect is a Gnome alternative to KDE Connect that also uses the KDE Connect protocol. Although it uses the KDE Connect protocol, GSConnect doesn't depend on the KDE Connect desktop app and, in fact, it doesn't work with the KDE Connect desktop app.

Installing GSConnect isn't as direct as the steps for installing other apps from the Ubuntu store. You need a Gnome extension tool on your browser, which is probably already installed. Chrome or Firefox both work well.

Open Gnome Extensions on your browser [6], search for GSConnect, and install the extension. The next step is to configure GSConnect on your Linux system. First, connect your Android device and Linux system to the same network. Then, on the top-right bar, click the arrow on the GSConnect option (Figure 8).

Click the *Mobile Settings* option and a new window pops up on your Linux system showing the ID for your Android device with the label "unpaired" (Figure 9). Similarly, you can see your Linux PC name on the KDE connect mobile app. You can initiate pairing from either of your devices. For now, I'll use the Linux desktop app. Click on the device name, and then click the *Pair* button on the top side of the window (Figure 10).

Now head to your Android device and accept the pairing request as you did in the case of KDE Connect. Once you've paired the devices, you can see your device status from the top bar of the Gnome settings (Figure 11). Finally, you're all set to control your Android device.

Conclusion

Controlling your Android device from Linux isn't just possible – it is downright powerful. Tools like ADB, Scrcpy, KDE Connect, and GSConnect simplify your access and control over Android devices. Start with GSConnect and then graduate to Scrcpy. The tools described in this article are useful for presentations, tutorials, or any other task that would benefit from controlling your Android device.

Of course, you'll need to stay safe and proceed cautiously. The box entitled "Keep It Safe" describes some best practices for keeping your environment secure. The "Troubleshooting" box offers for some tips on what to try if you experience problems. Don't worry if you feel a few hiccups; once you've got it running, it is smooth sailing. ■■■

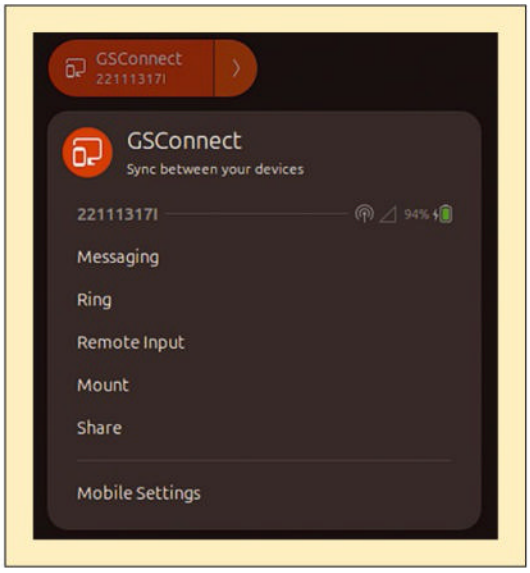


Figure 11: Device control options in Gnome top bar settings.

Keep It Safe

Although USB debugging opens the door for advanced functionality, it's essential to be mindful of the risks and follow best practices to minimize exposure. For example:

- **Only enable USB debugging when you need it.**
- **Always verify the RSA fingerprint when connecting.**
- **Keep your tools updated.**
- **Never approve unknown debugging connections.**

Troubleshooting

If things go sideways, try the following:

- Cable issues? Try a different USB port and cable.
- Device not showing up in ADB? Run the `kill-server` command described earlier in this article.
- Enable and disable the USB debugging option alternatively.

Info

[1] Android Debug Bridge (ADB) Documentation: *https://developer.android.com/tools/adb*
[2] USB Device Listing Command (lsusb): *https://linux.die.net/man/8/lsusb*
[3] Introduction to Udev Rules: *https://wiki.debian.org/udev*
[4] Scrcpy: *https://github.com/Genymobile/scrcpy*
[5] KDE Connect: *https://kdeconnect.kde.org/*
[6] GSConnect Gnome Extension: *https://extensions.gnome.org/extension/1319/gsconnect/*

The Author

Ali Imran Nagori is a technical writer and Linux enthusiast who loves to write about Linux system administration and related technologies. He blogs at *tecofers.com*. You can connect with him on LinkedIn.

■■■

Transfer data on heterogeneous intranets with LANDrop

Delivery Service

It does not always have to be Samba. LANDrop lets you share your data on the local network without having to go through a server.

BY ERIK BÄRWALDT

The fast data exchange of images, music, and videos without intermediate servers and without registration is a feature that is particularly popular on smartphones. But programs such as Apple's AirDrop or Google's Quick Share only interact with devices on their own platforms.

This is where LANDrop [1] steps into the breach. The free software supports all popular platforms and enables uncomplicated and fast data transfer between PCs and smartphones in a peer-to-peer (P2P) setup without intermediaries.

Technology and Setup

The P2P solution for data transfer on the local network is available for all the popular operating systems. Even some TV sets can be used for data transfer via LANDrop. The program only works with devices that are on the same network. Data transmission is encrypted. In contrast to other applications of this type, the software does not compress videos and image data.

The program is available as an AppImage package for Linux, which means that it can be used on all of the major distributions [2]. After downloading the package, which weighs in at around 32MB, assign execute rights with the first terminal command from Listing 1. Then launch the software with the second command or click on it in the file browser.

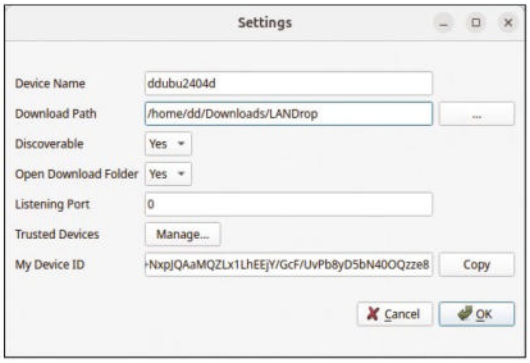


Figure 1: The LANDrop configuration dialog is limited to the essentials.

Alternatively, add the application to the menu system using the AppImageLauncher [3] tool. LANDrop then adds itself to your desktop environment's menu and automatically updates (if this is supported) when new versions are released.

Interface

LANDrop does not open a window at startup but simply drops an icon into the desktop environment's system tray. Right-click on the icon to open a context menu where you can initiate the most important actions.

After the install, I recommend that you first configure the application by going to the *Settings* item in the context menu. In the new window, enter the desired device name and specify a path in which the app will save downloaded data. You can also specify whether your computer will be visible to other LANDrop instances on the intranet. Additionally, you can define whether to open the download folder automatically after downloading a file (Figure 1). If the folder does not yet exist, the routine creates it automatically when the window is closed.

You can then start sending content. LANDrop offers two dialogs that are visually almost identical. You can click on the *Send File(s)* option in the context menu to open the window for sending files and folders. In the list area at the top, select files or directories for transfer either by drag and drop or using the *Files* and *Folder* buttons. The *Remove* button to the right removes selected data from the transfer list.

The lower window segment shows all the devices on the intranet that LANDrop has identified as potential targets. Click on one of these devices to select it as the target and then send the data to the remote device by pressing *Send* (Figure 2).

Listing 1: Starting LANDrop

```
01 $ chmod +x LANDrop-latest-linux.AppImage
02 $ ./LANDrop-latest-linux.AppImage
```

A notification then appears on the receiving device indicating the number and size of the incoming data. The recipient can decide whether to accept the transfer. If they do so, the app saves the data in the specified target directory. The software displays a progress indicator on both the sending and the receiving devices. Once the data transfer is complete, the app opens the target directory with the transferred data on the target computer. When you transfer complete folder hierarchies, their structure is kept.

Chatty

To send text messages with LANDrop, select the *Send Text* option from the context menu. In the new window, paste text from the clipboard into the input field; you can use Ctrl+V or press the *Paste* button to do this. Alternatively, type the message directly into the box. Then specify the recipient of the message in the area at the bottom and press *Send* to do precisely that.

A text window with the content of the message then opens on the target side. The window also shows the sender. They can use the *Copy* button to copy the content to the clipboard.

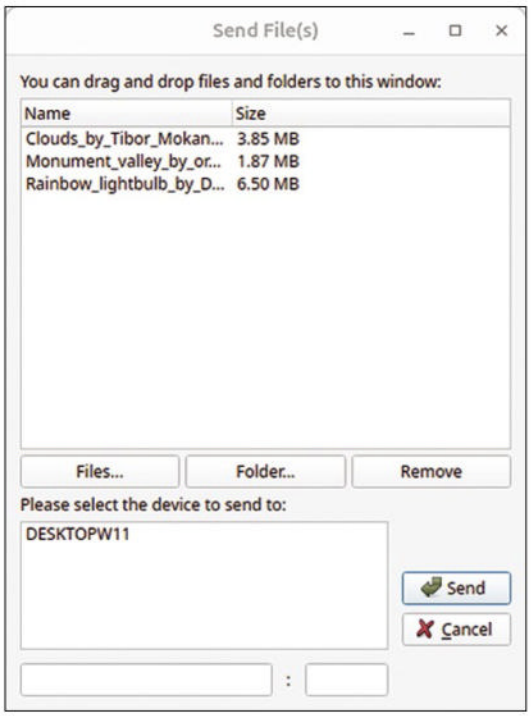


Figure 2: You can send files and folders to a device with just a few clicks.

Please note that LANDrop does not save the messages you receive. And there is no reply option in the window. Closing the text window deletes the message. In other words, this function is not a drop-in replacement for an instant messenger, but instead useful for notifying the recipient about sent content.

Compatibility

Because LANDrop is under active development, new versions are released regularly. Select the *Check for Updates* entry to check for new versions. If an update is available, you cannot trigger it directly from here. Instead, the routine opens the download page of the project from where you can pick up the new version.

Currently, automatic updates cannot be configured in the application's configuration dialog either, which means that you need to download any new variants as AppImages. Older versions of LANDrop are compatible with newer versions; this means that transfers between computers with different versions is possible.

Conclusions

LANDrop makes data transfer in heterogeneous intranets child's play. It also lets you involve mobile devices such as smartphones without having to use a server. LANDrop's only minor weakness right now is the need to integrate the AppImage package manually. On the positive side, LANDrop has a largely self-explanatory interface that makes it easy to get started. ■■■

Info

[1] LANDrop: *https://landrop.app*

[2] LANDrop download (scroll down): *https://landrop.app*

[3] AppImageLauncher: *https://github.com/TheAssassin/AppImageLauncher*

The Author

Erik Bärwaldt is a self-employed IT admin and technical author living in United Kingdom. He writes for several IT magazines.

Investigate the Linux engine beneath the Steam Deck's glossy chassis

Explore!

The Steam Deck game console runs on SteamOS, a variant of Arch Linux. While you typically use the Steam Deck like any other dedicated game machine, you can freely drop into a Linux Desktop mode of surprising and powerful potential.

BY JASON MCINTOSH

The Steam Deck, the versatile game console that Valve began selling in 2022, is hardly the first consumer-oriented technology that runs Linux under the hood. Readers of this magazine might have fond memories of the original TiVo boxes, for example, or have noticed a full copy of the GPL lurking in the deepest recesses of their cars' dashboard UI.

However, the Steam Deck stands apart in the level of transparency it offers regarding its Linux underpinnings. While its primary user interface is optimized for non-windowed displays and game-controller navigation, the Steam Deck explicitly invites users to alternatively use the device as a self-contained Linux machine running a KDE Plasma desktop. Bolder Steam Deck owners can use the power of its underlying operating system to extend and enhance their gaming experience.

This article presents an introductory guide through the user-explorable parts of SteamOS. I aim to empower you to playfully investigate this less obvious Steam Deck mode while still maintaining the core intent of the device: playing video games on a healthy and up-to-date SteamOS. More radical hacks of the Steam Deck hardware are certainly possible – as with any PC, you can wipe it and install your own Linux on it, if you want – but that is beyond the scope of this article.

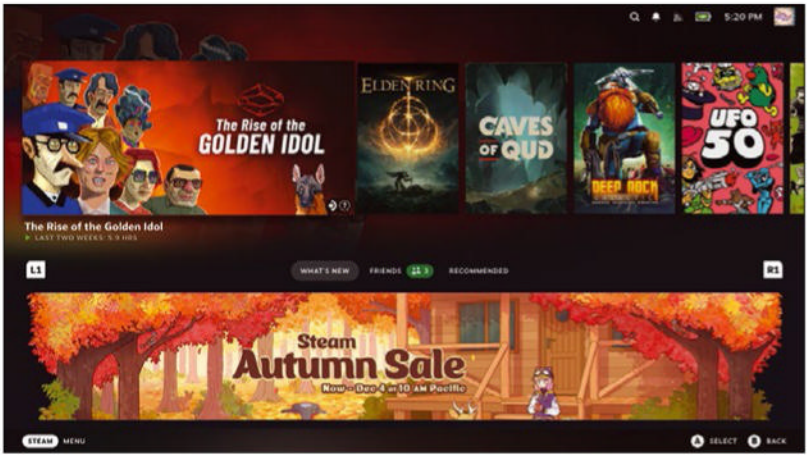


Figure 1: A typical view of the Steam Deck Gaming mode, right after turning on the console.

SteamOS in a Nutshell

The Steam Deck runs SteamOS 3, a derivative of Arch Linux. (Versions 1 and 2 of SteamOS were based on Debian and developed for Valve's previous experiments with commercial gaming hardware in the 2010s.) SteamOS 3 adds a handful of other Valve-developed technologies specific to the Steam Deck, such as the Proton compatibility layer that lets the machine efficiently run games and other software designed for Windows.

SteamOS divides its user interface into two major modes called *Gaming mode* and *Desktop mode*.

Gaming mode presents a user experience that resembles that of any other modern, Internet-connected game console, such as a Nintendo Switch or a Sony PlayStation. Gaming mode is designed to be operated entirely through the game-controller hardware built into the Steam Deck itself, or through a separate, Bluetooth-connected game controller that you provide. The Gaming mode UI is optimized for a single, non-windowed display, such as the Steam Deck's built-in screen or a connected TV. This is the default operation mode for the Steam Deck.

Figure 1 shows my Steam Deck in Gaming mode with big, colorful panels representing a handful of games I have installed – as well as an enticement to buy more.

Desktop mode is, very literally, a stock KDE Plasma desktop environment. It ships with a bit of Steam Deck branding and shortcuts to help you get your bearings, but you are otherwise free to treat it like any other Linux desktop. You can install non-Steam software using the command line or the KDE Discover application. You can even fire up Python and write your own software or stick something weird in your gaming machine's systemd timers!

Figure 2 shows my Steam Deck in Desktop mode, running the Steam desktop application, as well as a terminal window and a web browser.

Alongside this flexibility, the Steam Deck is designed to operate like a remotely managed device, much like other modern game consoles.

Figure 2: The same Steam Deck seen in Figure 1, only now running in Desktop mode.

While you can explore and modify the default user directory and a few other filesystem locations, SteamOS locks significant portions of the filesystem behind a read-only wall that not even `sudo` can breach. This prevents you from making changes that might get unexpectedly overwritten the next time Valve pushes an OS update to your device or which might prevent that update from running at all. Still, particularly risk-tolerant Steam Deck users can disable this safety feature if desired, as described later in this article.

Reasons to Pop the Hood

Even if you intend to use the Steam Deck only for playing games, bringing your familiarity with Linux to the device can significantly expand its potential to help you find fun and creativity.

Install and Manage Mods

A primary advantage that the world of PC gaming has over most consoles is the rich culture of game mods, player-developed modifications for game behaviors. The effects of mods can span the gamut from minor rules tweaks to entirely new fan-made works built from existing games' engines.

The combination of a user-accessibile filesystem and the Proton compatibility layer gives Steam Deck players access to game mods, even ones designed for Windows. The way that you install and manage mods varies among games. Some more polished games feature built-in mod-management libraries, while scrappier titles might require you to get far more manual, firing up a terminal window to make painstaking changes to data files or directories. In the latter case, the Desktop mode gives you all the power you need.

Install Games from Other Sources

Gaming mode is optimized to let you install, play, and – let's be honest – purchase games distributed through Steam. However, you can make almost any Linux or Windows executable installed on your Steam Deck available through Gaming mode. These can include manually installed applications, platform emulators like RetroDECK [1] or EmuDeck [2], or entire alternate games ecosystems such as Heroic Launcher [3]. In any case, doing this requires a side trip through Desktop mode.

As a simple example, you can use the KDE Discover application in Desktop mode to find and install SuperTux 2, an open source game similar to Super Mario Bros. You can then play it right away in Desktop mode, using a connected game controller. If you want faster access to the game in the future, then you can use the Steam desktop application to add SuperTux 2 to your Steam library, which lets you run the game from Gaming mode.

Tinker with Your Game Files

Because all of your games install their files in the user-writeable part of the filesystem, you are free to explore and modify these files however you like, using tools and techniques you're already well familiar with. Want to use `cp -av` to back up a game's data file and then see what happens to the game if you open the original in `vi` and change a thing or two? Nothing's stopping you!

Necessity can also be a motivator, beyond curiosity. Recently, I used my Steam Deck to play The Invincible, an excellent video game adaptation of the Stanisław Lem novel. My enjoyment of the game was briefly threatened by a disaster probably familiar to anyone with more than a little Linux experience. An update to the game collided with my

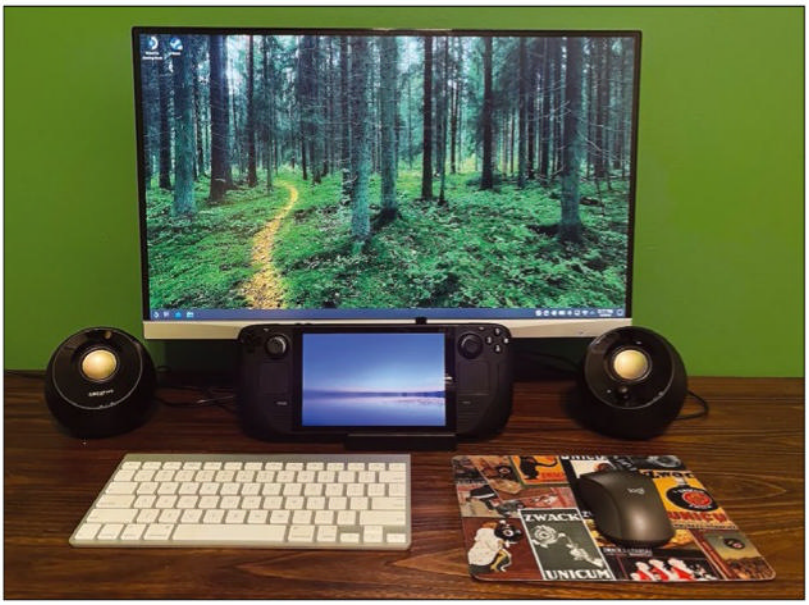


Figure 3: The author's preferred arrangement for using the Steam Deck in Desktop mode: docked, acting as a secondary screen to an external display, and with a keyboard and mouse connected over Bluetooth.

overstuffed Steam Deck running out of storage, resulting in my saved-game file being corrupted.

But, thanks to my Linux knowledge – and my experience at tiptoeing through much more serious disasters over a long engineering career – I was able to recover my saves through research, exploration, and careful experimentation. In the end, I used `touch` to create an empty file that the game's logs told me it expected to find in a certain location, and I was able to resume my perilous mission on the planet Regis III, curled comfortably on my couch wearing headphones.

This adventure, while uninvited, still left me feeling pretty good about my own diagnostic skills – as well as the power and flexibility of the Steam Deck. I dare say I wouldn't have been able to accomplish this sort of rescue on my PlayStation 5.

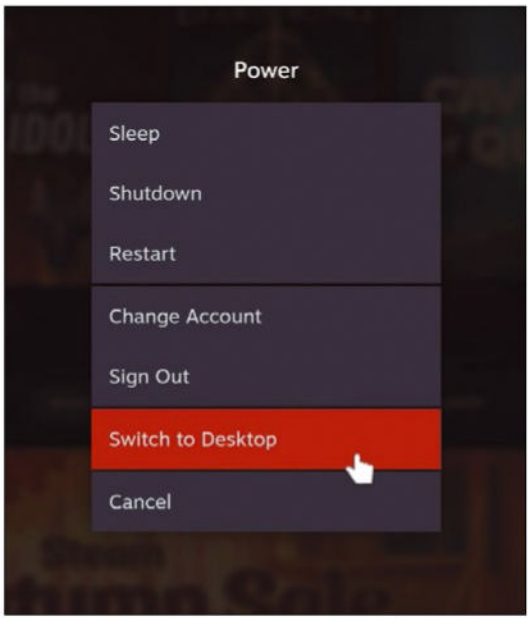


Figure 4: It's not the most obvious menu option, but it's not hidden, either.

Setting Up Desktop Peripherals

You can use Desktop mode without connecting external input devices or displays to your Steam Deck. Its built-in display can fit the whole KDE desktop, albeit at a tiny scale that might require some squinting. You can control the mouse pointer though the Steam Deck's on-board joysticks, trackpads, and triggers – or even the built-in touch screen, with a bit of patience. A virtual keyboard allows you to type using that same touch screen, similarly to how you enter text on a modern smartphone.

If you're in a pinch, being able to fall back to some or all of these measures can be helpful. However, if you want to explore the Linux side of your Steam Deck with comfort, efficiency, and minimum eye strain, then you need to connect it to a proper keyboard, video, and mouse setup.

The Steam Deck hardware includes both Bluetooth support and one USB-C port. This is enough to let you connect a wireless keyboard and mouse, for example, and then plug in a display using an HDMI-to-USB cable. However, this setup relies on battery power, because it takes up the sole USB-C port leaving you with no way to charge the device. This makes it inappropriate for anything other than very short-term use.

To let your Steam Deck really shine as a part-time Linux PC, use a peripheral that expands its USB-C port into a variety of other ports, including a route to power. While many solutions exist, I prefer Valve's own Steam Deck Docking Station. This little bit of custom-molded plastic and rubber includes three USB-A ports, HDMI and DisplayPort outputs, and a power pass-through. It also holds your Steam Deck in a propped-up, screen-forward manner, allowing you to position it relative to an external display so that you can use Desktop mode with both the machine's built-in screen as well as the external display.

Figure 3 shows my own preferred Steam Deck desktop arrangement. The mix of a commodity Acer display, an ancient Apple Bluetooth keyboard, and a much newer Logitech Bluetooth mouse demonstrates how the Steam Deck is happy to work with a variety of standards-capable devices.

Because the Steam Deck is designed to let you pair almost any Bluetooth-using game controller – even ones associated with other game consoles – the system offers rich Bluetooth configuration UIs in both Gaming mode and Desktop mode.

Of the two options, Gaming mode offers a simpler UI for Bluetooth pairing – and one that's much more legible on the small Steam Deck screen, if you haven't set up an external display yet. While intended to let you connect game controllers, this UI lets you pair any Bluetooth devices in range, including keyboards and mice. For more information, see the Bluetooth section of the "Steam Deck – Basic Use & Troubleshooting Guide" [4].

Having traditional input devices ready to use with your Steam Deck can have beneficial side effects, beyond making Desktop mode easier to use. Hardware mice or keyboards that work in Desktop mode also work in Gaming mode. Some games, such as traditional "point-and-click" adventures, present situations where grabbing a physical mouse can provide a much more comfortable experience than pushing a pointer around with a gamepad.

Even if you don't plan on using the Linux desktop with your Steam Deck very often, the modest investment in pairing up a mouse and keyboard with it can prove worthwhile.

Launch Desktop Mode

The SteamOS Gaming mode hides the doorway to Desktop mode in plain sight as the next-to-last option on the Power menu. To enter Desktop mode from the default Gaming mode, follow these steps:

1. Press the *STEAM* button on the Steam Deck, or the home button on your Bluetooth-connected game controller. (The home button varies depending upon your controller. For example, on a PlayStation controller, it is the button with the PlayStation logo on it.)
2. Select *Power* from the SteamOS menu panel that appears to open the Power menu (Figure 4).
3. Select *Switch to Desktop*.

After a few moments, a KDE Plasma desktop appears. If you have already plugged in an external display, the desktop appears on both the Steam Deck screen and the external display – it probably looks a bit backwards from what you expected. The next section addresses a one-time fix you can apply to your display setup before you start exploring the desktop.

Make Your External Display Primary

The first time you use an external display with your Steam Deck in Desktop mode, the device probably treats the Steam Deck's built-in screen as the main display, with the other one as secondary. If this happens, then the KDE panel, desktop shortcut icons, and other main-display features all appear in tiny form on the Steam Deck screen, while the external display shows a much larger, entirely empty desktop background. Even if you have sharper eyes than mine, this is probably not what you want.

To permanently assign the external display as the primary desktop screen, follow these steps:

1. On the Steam Deck screen, in the KDE panel, click the Application Launcher. Its icon resembles a Steam Deck logo, located in the bottom-left corner of the screen.
2. In the Application Launcher panel, select *Settings* to open the System Settings window.
3. Select *Display and Monitor*.
4. From the Device menu, select the name of your external display. Generally speaking, this will be the device *not* named *Built-in Screen*, which is the identifier of the Steam Deck screen.
5. Select the *Primary* radio button.
6. Click *Apply*. The KDE panel and other desktop elements move from the Steam Deck screen to the external display.

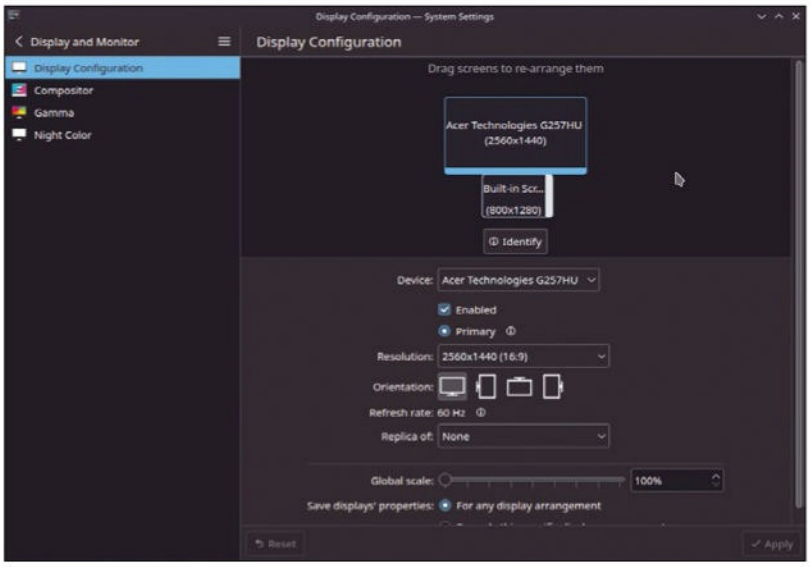


Figure 5: The author's preferred Display and Monitor settings for Desktop mode.

Your next step depends on whether you want to continue using your Steam Deck screen as a secondary display while in Desktop mode, as shown in Figure 3, or whether you want to keep Desktop mode contained entirely on the external display while it's connected.

To use your Steam Deck screen as a secondary display:

1. Reposition the two boxes in the display diagram so they reflect how you have positioned your Steam Deck and your external display relative to one another. Figure 5 shows what my Display and Monitor settings looks like, enabling the setup shown in Figure 3.
2. Click *Apply*.

To deactivate your Steam Deck screen while your external display is connected:

1. From the Device menu, select *Built-in Screen*.
2. Clear the *Enabled* checkbox.
3. Click *Apply*.

You can re-enable your Steam Deck screen at any time by following the previous instructions again, selecting the *Enabled* checkbox instead of clearing it.

SteamOS remembers your settings with this display so you don't need to repeat these steps the next time that you connect your Steam Deck to it. If you connect the Steam Deck to a different display, then you'll probably need to repeat this procedure.

Explore Desktop Mode

SteamOS Desktop mode logs you in as a non-privileged Linux user named `deck`. Its home directory is `/home/deck`, and you are free to use all the tools that KDE Plasma makes available – including the

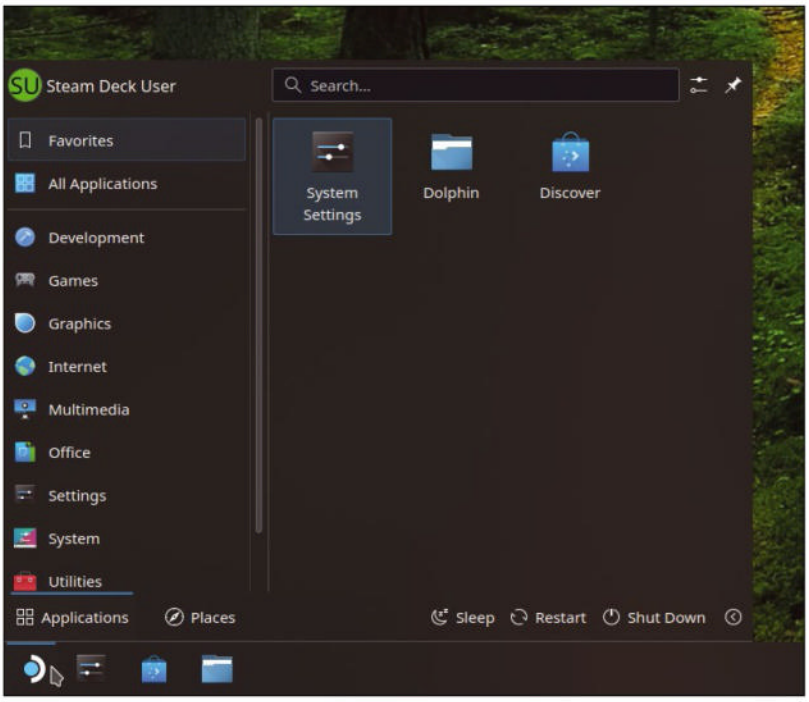


Figure 6: Clicking the Application Launcher, shaped like a Steam Deck logo.

Dolphin filesystem browser and Konsole terminal windows – to explore the system.

In normal use, all user-level operations you perform on Steam Deck are performed as the `deck` user, regardless of which Steam user accounts you have registered with Gaming mode.

For the sake of simplicity, I recommend using `deck` for all your Steam Deck exploration and experimentation. As noted later in this article, SteamOS does give you tools to grant `deck` more privileges when needed.

The first Desktop mode UI element to get familiar with is the KDE panel, which runs across the bottom of the primary display by default. The first icon on it is the Application Launcher, which in SteamOS is shaped like a Steam Deck logo (Figure 6). (Don't confuse it with the *Return to Gaming Mode* desktop shortcut, which is *also* shaped like a Steam Deck logo. I cover that later in this article.)

Treat the Application Launcher as your home base while you become familiar with Desktop mode. You can use it to find and run every tool and application mentioned in this article.

Aside from the filesystem locks described earlier, this really is a fully functional desktop Linux environment. You can use it with the same expectation of safety that you would find on any other Linux machine, when logged in as an unprivileged user. That is: As long as you understand what you're doing, you're unlikely to catastrophically delete your data or otherwise mess up the operation of your Steam Deck by accident.

When you feel bold enough to deactivate some of these safety measures, see the Advanced Use section later in this article. Until then, you should feel free to explore! Desktop mode gives you a great opportunity to get familiar with KDE Plasma.

To launch the Dolphin file browser, click the Application Launcher and select *System* | *Dolphin*. This opens a window focusing on part of the `deck` user's home directory.

When you are ready for a Linux command line, click the Application Launcher and select *System* | *Konsole*, as shown in Figure 7. And there you have it: a Linux command line on your video game console, with minimal resistance on the console's part.

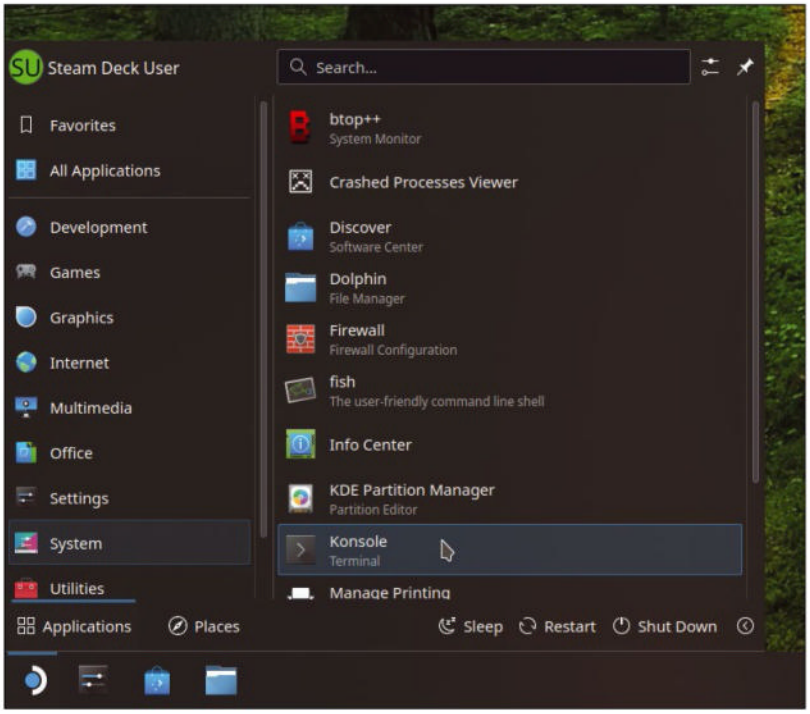


Figure 7: How to summon a command-line terminal in Desktop mode.

Install with Flatpak

The package manager of choice with SteamOS is Flatpak. Because SteamOS allows you to read and write to the `/var/` directory, Flatpak can install software in its usual default location of `/var/lib/flatpak`. This means that you can use Flatpak right away without running into any complications stemming from the read-only parts of the SteamOS filesystem.

The easiest way to explore available Flatpak packages is the Discover application. To use it, click the Application Launcher and select *System* | *Discover*. The window that opens lets you browse for software available from the Flathub application repository, all of which is just a click away.

For your first Flatpak foray, consider installing a web browser! The Steam Deck doesn't ship with any web browsers preinstalled – but Chrome, Firefox, and a bevy of other familiar favorites are all readily available through Discover. Even if you're unlikely to use your Steam Deck for day-to-day web access, you'll likely find a browser handy to have for one reason or another as you continue your explorations of the device's Linux side.

For more versatile use of Flatpak, you can run the `flatpak` command-line program from a terminal window (see [5] for more information).

Don't Use Pacman

Avoid using Pacman to install software on SteamOS – despite the fact that Pacman is the typical package manager for Arch Linux, the distribution from which SteamOS is derived. Pacman packages generally expect that the root user can freely modify any part of the filesystem – something SteamOS discourages by putting most of the filesystem behind a read-only lock. While you can disable this lock and install software using Pacman anyway, doing so runs the risk of unpredictable behavior after subsequent SteamOS updates, for reasons described in the next section.

To help keep your Steam Deck healthy and up-to-date, restrict yourself using the Flatpak package manager.

Advanced Use

SteamOS puts two safety layers between the factory-default state of the `deck` user and unfettered access to the entire system.

First, you can let `deck` run superuser commands by assigning it a password. To do this, run the `passwd` command in a terminal window and supply a password when prompted. From then on you can have the `deck` user run commands as the superuser by using `sudo`, just as you would on a typical Linux system. To remove this ability, remove the `deck` user's password by running `passwd -d`.

For complete access to the filesystem, including the parts that SteamOS normally locks away behind a read-only barrier, run

```
sudo steamos-readonly disable
```

I don't recommend doing this casually, because it lets you accidentally mess up the core operation of your Steam Deck – or even expose it to remote attacks by clever malefactors. At the very least, files that you write to the normally read-only directories risk being quietly overwritten the next time SteamOS downloads and applies an OS update image.

In all of my Steam Deck experimentation, I've never felt the need to flip the `steamos-readonly` switch. But, if you ever really need it, it's there. If you do flip it, you can always reset it with

```
sudo steamos-readonly enable
```

Exit Desktop Mode

To return to gaming mode, perform any one of the following actions:

- Open the *Return to Gaming Mode* icon on the desktop, as shown in Figure 8.

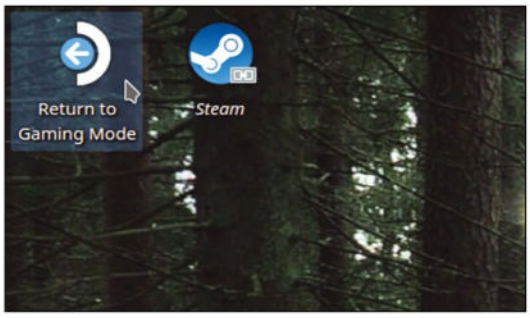


Figure 8: Get me out of here!

- Restart the Steam Deck.
- Log out.

After a moment, the full-screen, console-style Gaming mode returns to the external display, if it's still connected, or to the Steam Deck built-in screen.

Happy Hacking!

Among game consoles, and even among consumer hardware in general, the Steam Deck offers an unusual amount of trust and transparency to its users. Its Desktop mode presents an environment that invites curiosity, exploration, and tinkering among Linux users of all skill levels, operating in the interestingly purpose-built environment of a game console rather than a general-purpose PC. Even experienced Linux users who love gaming will likely find new inspiration from exploring the power and potential granted by the surprisingly exposed inner workings of SteamOS.

If you own a Steam Deck but haven't popped its hood yet, then I hope this article inspires you to explore. For more information, see Valve's FAQ about the Steam Deck Desktop mode [6]. ■■■

Info

[1] RetroDECK: *https://retrodeck.net/*
[2] EmuDeck: *https://emudeck.com/*
[3] Heroic Launcher: *https://heroicgameslauncher.com*
[4] "Steam Deck – Basic Use & Troubleshooting Guide": *https://help.steampowered.com/en/faqs/view/69E3-14AF-9764-4C28*
[5] Flatpak: *https://docs.flatpak.org/en/latest/using-flatpak.html*
[6] Valve Desktop mode FAQ: *https://help.steampowered.com/en/faqs/view/671A-4453-E8D2-323C*

The Author

Jason McIntosh is a writer who lives in New York City. His personal website is *https://jmac.org*. You can also find *Venthuffer*, his audio zine about the Steam Deck, at *https://venthuffer.com*.

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

Nate explores the top FOSS including the latest Xfce desktop, an audio testing tool, a turn-based tank game, and an app for securely sharing secret messages and files. BY NATE DRAKE

Open and Shut

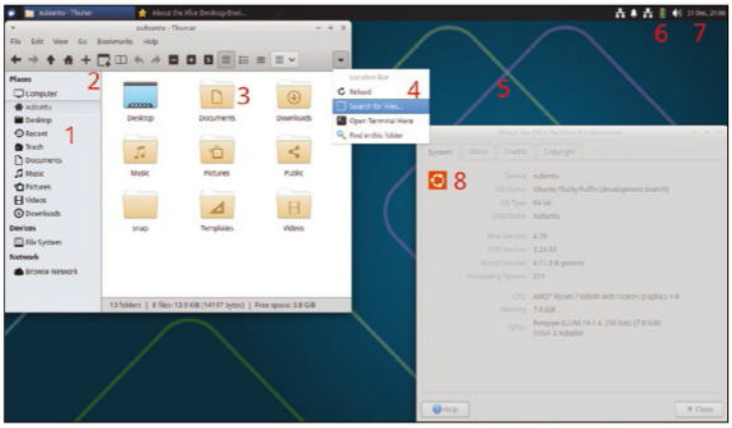
Nothing breaks my heart more than to see an open source project go behind closed doors. This was the case for AI Dungeon in 2019. Based on OpenAI's GPT-2, players can enter simulated worlds and input various text commands to interact with items and NPCs. It isn't perfect – for instance, the cyberpunk adventure I played made it clear that cowering inside a bathroom wasn't an option. Still, the game's runaway success meant the developers decided to turn it into proprietary software.

In this sense, AI Dungeon was a victim of its own success. The same can't be said of the startup Embodied, Inc., who created Moxie, an emotional support robotic toy for children. The company recently announced it's going out of business, and given that Moxie works over the cloud, the robot would be going with it. After facing customer backlash, CEO Paolo Pirjanian posted on LinkedIn to say that Embodied's technical team was working on an open source solution. This is in the form of a local server application, OpenMoxie, which could run on customers' computers. I can't help but wonder if they'd adopted this model in the first place, would they still be going out of business? Manufacturers, take note.

Desktop environment

Xfce 4.20

Although Gnome and KDE receive the most attention, Xfce remains a firm favorite with Linux users for its lightweight and modular approach. The recent release of version 4.20 is accompanied by an epic changelog, detailing various bug fixes. I encourage readers to go through it, because some exciting changes are afoot in the Xfce camp. At the time of writing, the latest version hasn't been included in the stable release of any distro I could find, so I tested it using a daily build of Xubuntu 25.04.



1. **Symbolic icon:** Thunar's sidebar can now use symbolic icons. Other icon sets are unaffected. 2. **New toolbar icons:** Options include adding new file manager tabs or even opening new windows. 3. **Drag and drop:** Users who drag and drop files over folders can now open them automatically. 4. **Overflow menu:** If you add too many toolbar buttons to a window, they will be listed here. 5. **Wayland support:** While most Xfce components support Wayland, you'll need a compositor to test it. 6. **Power settings:** Hybrid sleep mode and energy rates for devices are now supported. 7. **Upgraded clock:** The digital clock can display week numbers. There's also a 24-hour analog clock. 8. **About page:** This now lists the OS icon, as well as the current windowing system and GPU.

Almost all Xfce components now support Wayland. A new library (*libxfce4windowing*) has been introduced for this purpose, though you'll need a wlroots compositor like labwc if you want to test this effectively. The Thunar file manager has also undergone some important upgrades. For instance, folders will now open automatically when a user drags a file onto them. The tool bar also new contains new options (accessible via the *View* | *Configure* toolbar). These include options to create a new folder tab or even a new window. If you add more toolbar buttons than can fit in the current window, Thunar also now displays a small overflow submenu to list them. The sidebar can also now display more subtle "symbolic" icons for key "Places."

The system's About dialog has also been overhauled. It now displays not only basic information about the OS and desktop environment but also the logo for the distro in question, as well as its current windowing system and GPU. The power mode has also been upgraded to support hybrid sleep mode. The energy rate for certain battery devices also now appears in the details tab. Despite the litany of upgrades, the Xubuntu daily build consumed only 1.1GB of RAM in my virtual machine.

Project Website

https://www.xfce.org/

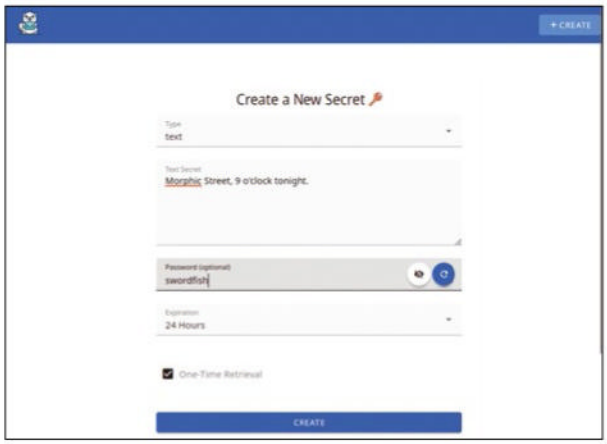
Secure secret-sharing utility

GopherDrop

Users of the freemium password manager Bitwarden may already be aware of its Send service. This ostensibly exists to make it easy to share messages and files via a secure link. Users can even choose to password protect this secret information or have the link expire after a set period of time. This serves as the inspiration for GopherDrop, which is a self-hostable REST API and UI for sharing encrypted one-time secrets and files. The GitHub page contains instructions for simple deployment in Docker, as well as a free online demo, which I used for the purposes of this review.

On first launch I noted the interface, written in Vue.js and Vuetify, lives up to the developer's claim of providing a "modern user experience." The fields are clearly laid out and are self explanatory. For instance, you can use the first drop-down menu to choose whether to protect a text message or a specific file. If you chose text message as I did, there's a field to enter your message. Otherwise GopherDrop displays a file dialog to select your chosen file. From examining the code, I'm not clear if there's any upper limit on the size of the file you upload. The same seems to be true for text messages, because GopherDrop happily generated a link for my Project Gutenberg copy of *The Adventures of Sherlock Holmes*, which runs to 107,553 words.

Once you've chosen the secret to protect, you can optionally set a password. You can also set an expiry date for the link (e.g., 24 hours), as well as choose *One-Time Retrieval*. After experimenting with this feature, I was pleased to see that once a message/file has been retrieved, GopherDrop says the link is invalid or the secret is expired. In other words, it's impossible to prove a link was valid after the fact without access to your server.



GopherDrop is designed for self-hosting files securely. Passwords, URL expiration, and one-time links are also supported.

Project Website
https://github.com/kek-Sec/GopherDrop

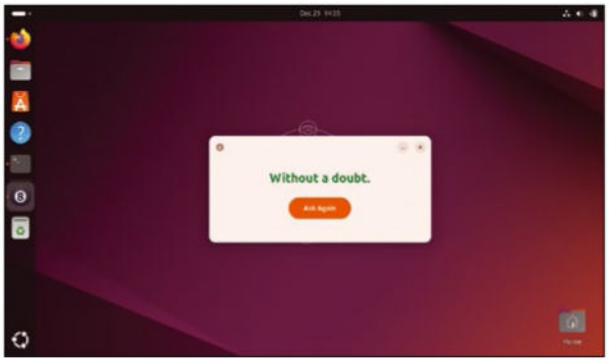
Magic 8 Ball simulator

Clairvoyant

Despite my long-standing skepticism of the occult, one of my very first programming successes was a simulation of the ever-popular Magic 8 Ball toy. This was in the dark days when I had a desk job and was tinkering with Visual Basic (VB) in Microsoft Outlook to try to "optimize my workflow." For those who've never played with this toy, owners can ask it an objective yes/no question and then turn it upside down to view the answer. The ball contains a 20-sided icosahedron die suspended in liquid. Each face of the die has a different answer: 10 positive, five neutral, and five negative.

Naturally, writing a script that can show one of 20 responses at random is simple. However, developer Cassidy James Blaede has done so with such elegance that Clairvoyant has become an officially recognized Gnome Circle app. Official binaries are currently only available via Flathub. There is a version in the AppCenter in elementary OS but it's no longer maintained. The latest release has been updated to support Gnome 47's latest accent colors feature. It also has improved support for the Croatian language. Users can click *Ask again* to see a new response. From working through these, I see that Cassidy has tweaked the wording for some responses. For example, the stock response, "Cannot predict now" has been replaced with the Yoda-esque "Impossible to see the future is." Other innovative responses include "Naw" and "404 Answer Not Found."

From examining the project GitHub page, it appears that Clairvoyant is written in C and uses `/dev/urandom` to determine responses, meaning your system's principal entropy pool won't be affected by playing with it. As for my own ventures into VB, after programming the eight ball, I configured Outlook to scan emails for keywords and automatically send boilerplate responses. This reduced my workload to around an hour a day. Did I ever inform my employers of this? Outlook not so good.



Clairvoyant produces one of 20 random responses to objective (yes/no) questions in the same way as a Magic 8 Ball.

Project Website
https://flathub.org/apps/com.github.cassidyjames.clairvoyant

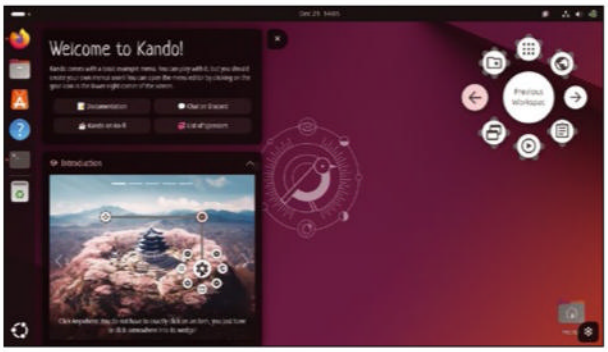
Pie menu desktop plugin

Kando

If you've watched some of the sci-fi greats such as *Minority Report* and the Iron Man series, you'll know that in the future all screens and head-up displays (HUDs) will use radial menus, often referred to as "pie" menus. To this end, Kando's purpose seems to be to offer a quick and easy way to access system files, web bookmarks, and apps from a compact and customizable interface. Binaries are available from Flathub, AUR, NixOS, AM, and AppMan. If you decide to give Kando a try, I strongly recommend visiting the dedicated Linux section in the online documentation (*https://kando.menu/installation-on-linux/*).

This was where I discovered that the Flatpak I'd installed wouldn't work out of the box with my Ubuntu 24.10 virtual machine (VM), because it runs Wayland. Instead, you need to install a Gnome Shell extension. On the plus side, the app also informed me of this upon launch and even helpfully supplied the correct link. Other desktop environments such as KDE are supported out of the box, though in some cases you'll need to install a compositor for transparency to work.

By default the app simply displays a small tray icon, but you can hold Ctrl+Enter to bring up an example menu. This contains links to frequently visited areas of your system such as Apps. Clicking into these can also open up pie submenus, as in the case of Apps, which listed top picks including the terminal and Gimp.



Kando has helpful instructions for creating your own pie menus. Click on the desktop and then hold Ctrl+Enter to invoke.

When you've finished exploring this, Kando also has helpful instructions on how to customize your own pie menu. You can create as many as you like, including those that appear only when certain conditions are met (e.g., only when a specific application is focused). Besides opening submenus, menu "wedges" can also be programmed to run a specific terminal command, open a URL in the default browser, simulate a macro or hotkey, and even paste text.

Project Website
https://flathub.org/apps/menu.kando.Kando

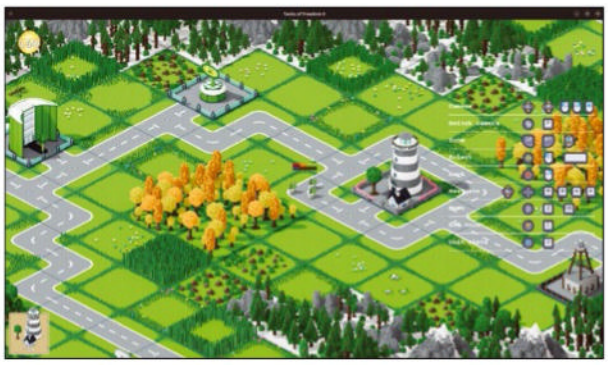
Turn-based strategy game

Tanks of Freedom II

This game is a sequel to the original Tanks of Freedom (TOF), the working prototype of which was built in 48 hours for Global Game Jam 2015. Version 2 has been crafted using the Godot engine. The developer also claims that all sound effects were created on real Game Boy hardware and that all voxels are hand drawn using a 32-color palette. The game is available as an executable from the main site. If you want to support the developer, you can also buy it for $5.99 from the Steam online store.

The premise of the game is that the player takes command of the army of one of four fictional nations in the game's world. For example, Ruby Dusk is an industrial nation. Its units and buildings have a clear steampunk theme and are colored in red. Like the original, TOF II also employs a simple resource management system: Players gain points for capturing assets such as enemy buildings, which can then be spent on new resources. This updated sequel has added a number of new units, including scout helicopters and rocket artillery.

I discovered this the hard way when I was unable to run the in-game tutorial for campaign mode. After opting for a skirmish against two enemy AI factions, I found my buildings were quickly overrun by enemy forces, including the updated infantry. When units defeat enemies, they also level up. This grants access to new skills to harass the enemy even further. As with many games of this kind, the best victory strategy seems to be to mix and match different units and skills. For instance, tanks can destroy any unarmored target. However, rocket artillery can pick them off at a distance while staying out of range. The game apparently has numerous biomes, though I only encountered the grasslands and desert during my short time with it. There's also a level editor for players to craft their own campaigns.



Players gain points for capturing enemy assets such as buildings, which they can then use to build new resources.

Project Website
https://czlowiekimadlo.itch.io/tanks-of-freedom-ii

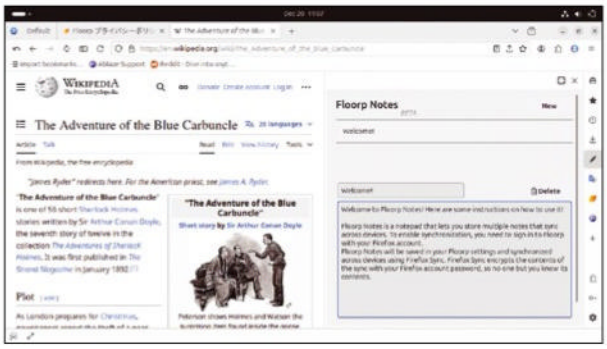
Web browser

Floorp

This Firefox ESR-based browser's main site claims that it offers excellent privacy and flexibility. As an InfoSec journalist, I'm always intrigued by products that claim to be the last word in online security. The browser is downloadable in compressed (TAR) format for both x86 and ARM-based Linux systems. After extracting and running the executable, Floorp displayed a friendly welcome tutorial offering to import data from my existing browser. The program also automatically opened tabs detailing the release notes in both English and Japanese.

Given that the main site claims the browser offers strong tracking protection, I was curious what Floorp has to offer above and beyond standard Firefox. After running the Electronic Frontier Foundation's Cover Your Tracks tool a few times, I noted that the browser seemed to have the same unique fingerprint. I also ran a few tests using *BrowserLeaks.com*, which showed that WebGL was disabled, presumably to frustrate fingerprinting. However, HTML5 Canvas code ran as normal, making it relatively easy for unscrupulous websites to track users. Similar privacy-focused Firefox forks such as LibreWolf also include a preinstalled ad blocker, so I was surprised to find no add-ons in Floorp. So much for privacy features.

However, Floorp does shine when it comes to its interface. During setup, users are offered a variety of themes. There's also a secondary sidebar for help with managing data such as tabs and bookmarks. The most recent version even supports a "floating" mode. The browser also has beta support for Notes, which users can store and then copy securely across devices using Firefox Sync. If, like me, you're constantly juggling multiple tabs, you'll be delighted by the Workspaces section for managing multiple sessions.



Floorp has extra features beyond standard Firefox, including a secondary Browser Manager sidebar and custom notes.

This is what really clinched it for me in recommending that readers give Floorp a try, because aspects such as bookmarks, tabs, and titlebars are highly customizable. Still, I'd recommend installing some privacy-friendly extensions such as uBlock Origin and Ghostery if you want to use this browser as a daily driver.

Project Website
https://floorp.app/en

AppImage launcher

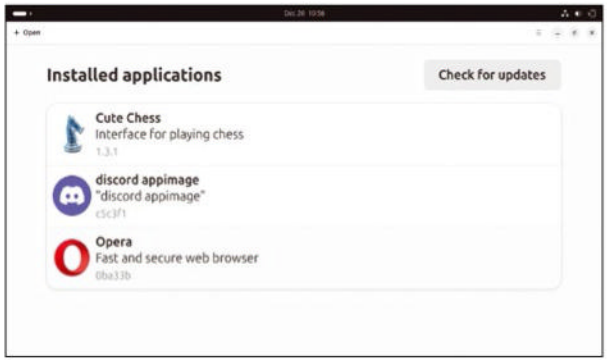
Gear Lever

Developer Lorenzo Paderi's output is truly prolific. Not only has he authored the Whisper audio testing app (more on that later), but also this utility for launching AppImages. While the format has been controversial with Linux diehards, there's no denying a gap in the market for binary Linux executables, similar to the `.exe` format in Windows. Gear Lever is currently available as a Flatpak. There's also an unofficial AppImage build. To use it effectively, you'll also need to install FUSE. On Debian-based systems such as Ubuntu, just run

```
sudo apt install libfuse2
```

On first launch, Gear Lever displays a helpful tutorial explaining how it works. From there, you can also open a folder containing a demo AppImage, so you can right-click and set Gear Lever as the default app. The utility's working directory is `~/AppImages`, but you can change this if you wish. Gear Lever's interface is very clean – users can either click the + or just drag and drop AppImage files directly into the window. By default, files aren't executable – you need to manually "unlock" them.

Clicking into an individual app displays a brief summary with *Launch* or *Remove* buttons. From here, you can also edit the metadata for the AppImage in question, as well as add a URL or command-line arguments. I took it upon myself to add an AppImage version of Discord and the game Cute Chess to test out the main window. This is where Gear Lever really comes into its own, because it displays all installed AppImages in a neat list. There's even a dedicated button to check for updates. This addresses a major flaw in AppImages, given that normally users must navigate to the folder containing the file and manually launch or update it on a case-by-case basis.



On my Ubuntu 24.10 VM, any AppImages I added to Gear Lever's app menu were also automatically listed in the System menu.

Project Website
https://github.com/mijorus/gearlever

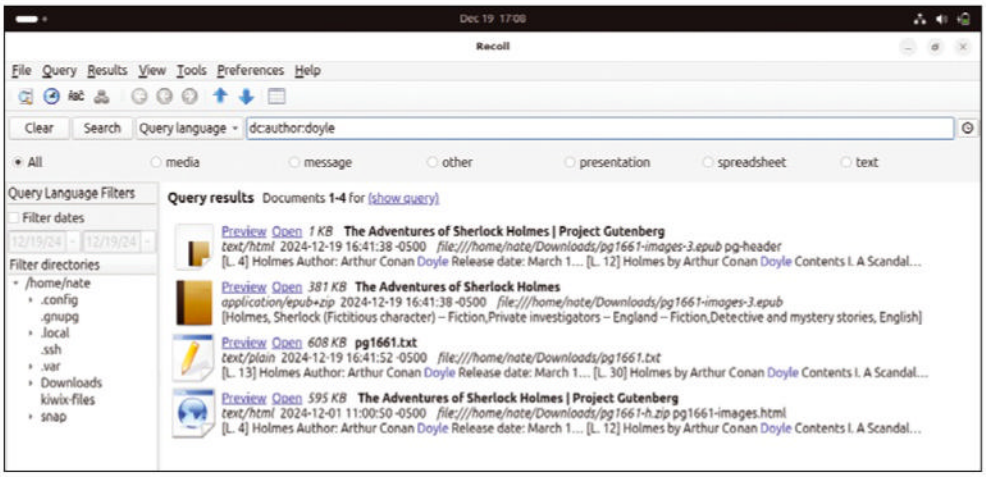
Search utility

Recoll

One thing I've always felt Linux lacked is a truly comprehensive search bar of the kind you find in macOS. I'm sure readers will tell me there's a specific Gnome extension or other module for your desktop of choice, but ideally searching files, content, and emails could be managed from a single place. Recoll by Jean-Francois Dockes aims to achieve this, as the main website claims it can find documents based on their content as well as filenames.

Although I opted for the Flatpak, Recoll exists in many distro repositories and as an Ubuntu PPA. There's also an AppImage, though the developer cautions that support for this is tentative. The interface is based on Qt; beneath the hood, it runs the very respectable Xapian search engine library. On first launch users are prompted to configure indexing. As the pop-up states, in most cases you can just click *Start Indexing Now* to begin. However, you can fine-tune settings to exclude specific directories, detail file paths, and so on.

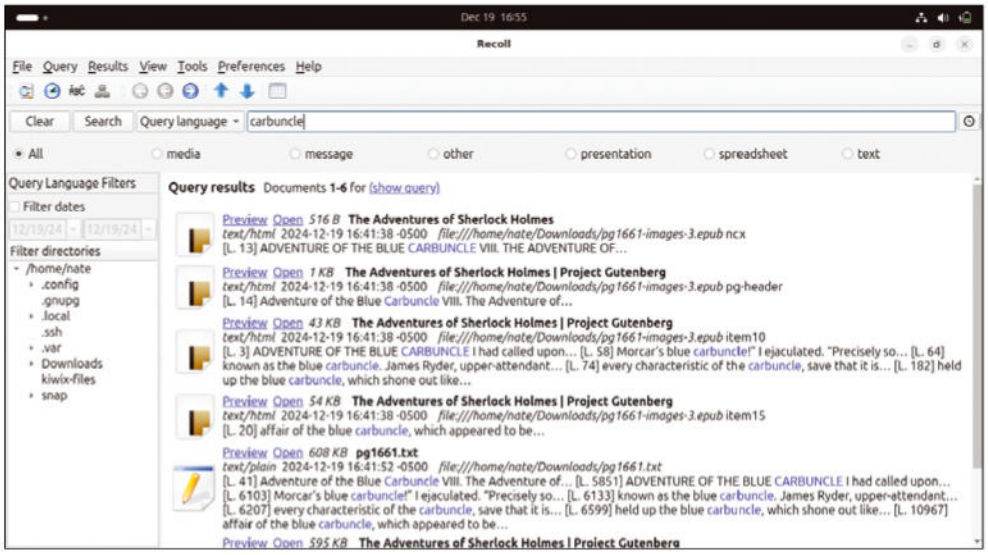
Given that the main website claims that Recoll can scan inside files and even archives, I decided to put it through its paces. First I downloaded a single track of Mozart from the Internet Archive. I also download a Mozart album in ZIP format, as well as the ebooks of *Dracula* and *The Adventures of Sherlock Holmes* in multiple formats. The main search bar defaults to "Query language." The helpful Recoll manual explains that this is based on the now defunct Xesam user search language specification. Naturally, you can just enter a search term such as "Sherlock Holmes" and search for files with titles or names that match that text. However, Recoll also supports searching specific fields. Users can enter a colon (:) to detail what this should contain (e.g., "author:doyle").



Recoll's query language also supports advanced searches such as by author name. You can even exclude specific terms.

When search results are listed, you can also opt to preview or open a file. However, Recoll doesn't automatically detect your default applications. This meant I encountered an error when trying to open a PDF, because I didn't have Evince installed. If multiple apps can open a file, you can choose which one to launch. For instance, when I chose to open an XML file both Firefox and Gnome's text editor were listed. I decided to push a little further with the search features by entering the term "carbuncle." The term was discovered in each of the Sherlock Holmes ebooks I downloaded, including those inside a ZIP archive.

For my final test, I installed Mozilla Thunderbird and ran a search for a specific phrase in an email attachment. This failed until I chose *File | Update index*, at which point it discovered the phrase in the email attachment almost instantly. I tried to start "real-time indexing" but encountered an error saying it was interrupted. This means I was stuck in batch mode. While I wish readers more success with the feature, it would definitely be a drag to have to update the index manually each time files and emails changed. However, overall this is a very easy-to-use utility capable of highly customizable searches.



You can use Recoll to run simple searches, such as text inside files. Click *Open* to view in the default application.

Project Website

https://www.recoll.org/

Image editor

AR Tag Placement Tool

AR Tag Placement Tool makes quick work of adding AR tags and QR codes to images. Its Finder Tool can also analyze existing tags.

At its core, this tool is designed for placing augmented reality (AR) tags, such as ArUco and AprilTags, as well as QR codes into images. As developer Bruce Belk explained on the /r/opensource subreddit (since deleted): "Measuring pixels or relying on approximations is frustrating and time-consuming, and can lead to inconsistent results."

I've reproduced his comments here faithfully, as I've no experience with this, although I've grappled with QR codes in the past which can be equally daunting. The source code for AR Tag Placement Tool is available from GitHub, as is a hosted web application of the tool. The interface is fairly bare bones but allowed me to easily browse to a picture of Tux I'd saved to my hard drive from Wikimedia Commons. There are options for adjusting the canvas background color and overall size. Once an image is loaded, you can alter its dimensions too.

The Marker section determines the particular type of tag or code to load into the image via the Dict drop-down menu. From here, you can set an idea as well as change the size or placement of the tag. When entering a QR code linking to *www.linux-magazine.com* I noted that you can also use the mouse to drag and drop tags elsewhere in the image. The Save section supports exporting your new, tagged image in PNG, JPG, or WebP formats.

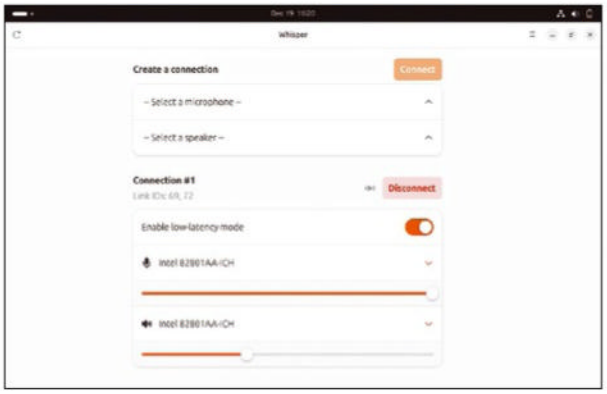
Special mention should also go to the Finder Tool, which can identify AR tags with unknown IDs. It can also experiment with different dictionaries, which you select via the dedicated drop-down menu, to understand marker patterns. It seems you then have to manually enter the marker by clicking on individual squares in a grid to render them black or white. It seems to me it might be simpler for the tool to support uploading an image of the existing marker. Still, the Finder Tool is simple to use and quickly identifies tags.

Project Website

https://github.com/BBelk/ARTagPlacementTool

Microphone testing

Whisper



Whisper allows you to select and connect audio inputs and outputs together to test your microphone settings.

If, like me, you're fond of listening to the sound of your own voice, then this Gnome app is for you. Whisper allows you to listen to audio input from your microphone through your speakers. The most obvious application for this is to play back your voice for testing purposes. In order to use the app, you must have both PulseAudio and PipeWire installed on your system. Developer Lorenzo Paderi helpfully reminds readers on his website that PipeWire is the default audio server for many mainstream distros including Fedora 34 and Ubuntu 22.04 or later. Binaries are only currently available via Flathub.

After installing the Flatpak, I took a moment to revel in the threadbare Gnome interface before selecting my laptop's internal microphone and speaker from the helpful dropdown menu. This then instantly forms a "connection" between audio inputs and sources. On the project page, the developer points out that this is done in a very similar way to Helvum – a graphical patchbay for PipeWire. However, the interface is simpler in that it only lists physical inputs and outputs, not virtual interfaces such as audio streams.

The latest version of Whisper (1.3.0) has also introduced a "low-latency" mode. This can impose a smaller buffer size for a specific device to reduce latency. However, Paderi cautions on the GitHub page that this can come at the expense of CPU usage and audio quality. For instance, it can cause audio pops if the system is under heavy load. However, during my tests with Whisper, I found my pear-shaped tones came through clearly whether low latency was enabled or not.

By default Whisper will release all current connections when the app is closed, though you can change this via the Preferences menu if you wish. From here you can also have the app display the connection ID – "for the geeks out there" – as well as enable Whisper upon system startup.

Project Website

https://flathub.org/apps/it.mijorus.whisper

Open source photo management

Online Gallery

Create, organize, and share great photo galleries online with the user-friendly, yet powerful, Piwigo.

BY MARCO FIORETTI

Today's scanners and smartphones make it very easy to amass thousands of photos on our computers, and free software such as digiKam makes it equally easy to manage and browse such collections on Linux systems. Desktop tools like digiKam, however, can only serve the single user who runs them. They cannot help people to build, catalog, and share a photo archive collaboratively through the Internet.

This tutorial explains how to solve this problem without locking your memories into anti-privacy walled garden such as Flickr, Instagram, or Google Photos. Piwigo [1], an open source photo gallery manager, runs on any web server or web hosting account with the right features. Piwigo's online photo galleries (as shown in Figures 1 and 2) are viewable in any browser, and users can browse, map, tag, and comment on Piwigo's galleries.

Indeed, all the images in these pages come from the test version of a real-world service I will soon put online for the 50th anniversary of a charity in my own neighborhood: a private (but web-accessible) archive of all the photos contributed from all members, in a way that makes it as easy as possible for them to rebuild and preserve the full story of their community (places, people, events, anecdotes, etc.) .

More exactly, I am going to show how I installed Piwigo on my Ubuntu 24 LTS desktop and tested how to configure and run it to upload photos, associate them to multiple albums, and make it possible for authorized users to browse and comment them, from any fixed or mobile device.

Installation: Long, Not Complicated

Piwigo is just one of countless LAMP applications – an acronym that indicates any software

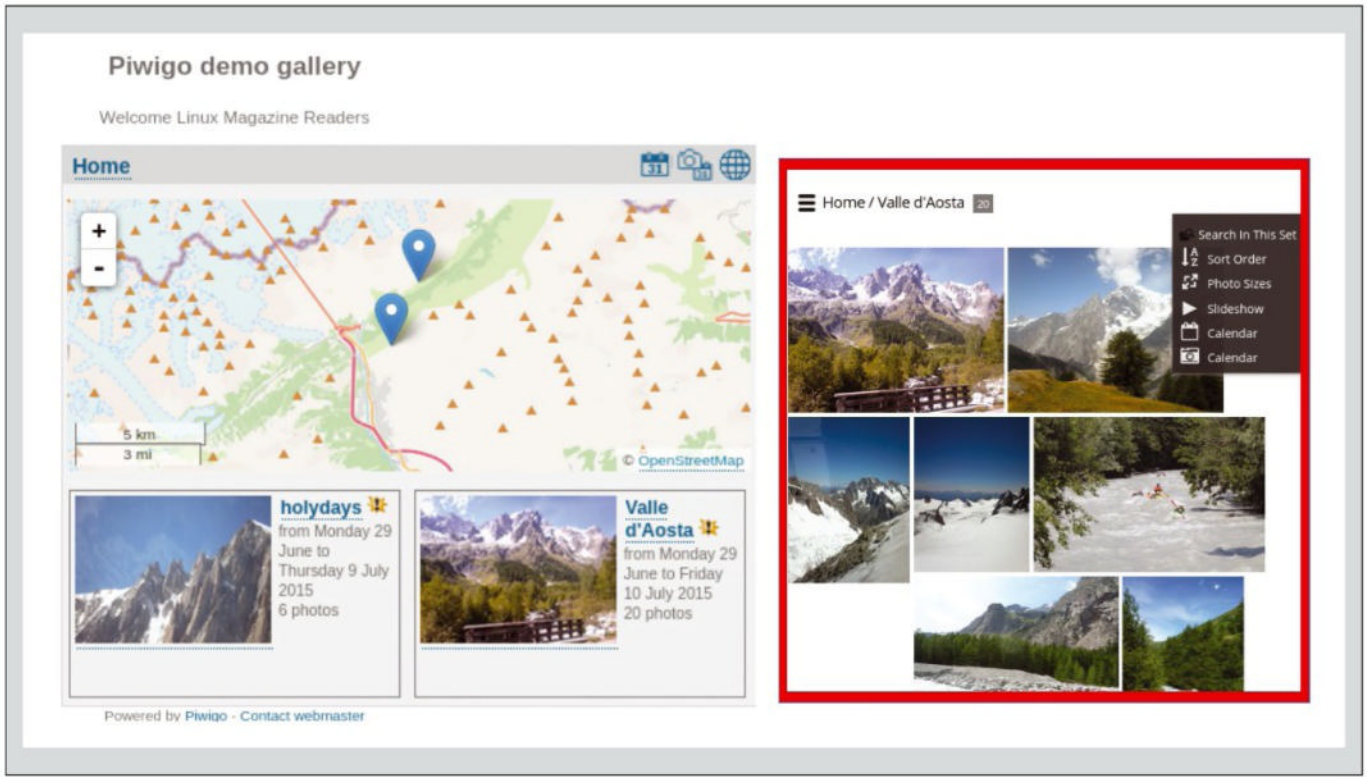


Figure 1: The homepage of a Piwigo installation showing all albums, a photo map (left), and the single album view (right).

written in the PHP (P) language that's loaded by an Apache web server (A) running on a Linux system (L) and that stores all its data in a MySQL or similar database (M). You can set up and run Piwigo on any web space account that includes access to compatible versions of Apache and MySQL or, as I describe below, install it and the whole the LAMP stack from scratch on any Linux server connected to the Internet. This takes some time and basic familiarity with the command line but is not difficult.

The first step (which is the only one for people using a hosted web space) is to download the Piwigo software archive from the website and uncompress and copy it on a directory of your Linux web server or web account. In both cases, you should also make the files usable by the actual web server. To do so on a hosted account, follow the corresponding instructions of your web space provider. On your own server, assuming that you installed all the files in a subdirectory of your web server root folder (`/var/www/html`) called `piwigo`, change the owner and group of all the Piwigo files with these two commands:

```
#> chown -R www-data:www-data /var/www/html/piwigo/
#> chgrp -R www-data:www-data /var/www/html/piwigo/
```

Next, the Packages

PHP applications are like shell scripts: They don't run by themselves but are executed by a separate executable "interpreter" which, in the LAMP case, is practically embedded into the Apache web server. Besides, for reasons outside the scope of this tutorial, most Linux distributions don't include the actual MySql server, but its clone, MariaDB.

Luckily, all these programs and all the auxiliary tools they need to make complex PHP applications such as Piwigo work are available as native packages in the standard, native repositories of the most popular Linux distributions. On Ubuntu 24 LTS, for example, all I had to do to make it ready for Piwigo was to run this command:

```
#> apt-get install apache2 mariadb-server php libapache2-mod-php php-common php-mbstring php-xmlrpc php-gd php-xml php-intl php-mysql php-cli php php-ldap php-zip php-curl unzip
```

(The actual command and set of needed packages, as well as their names, would be different on other distributions: For details, consult the Piwigo website!)

After installing those packages, it's necessary to prepare the MariaDB and Apache servers to support Piwigo. For MariaDB, this means running the script called `mysql_secure_installation`:

```
#> sudo mysql_secure_installation
```

Besides disabling certain features useful only for testing or in special scenarios, this script will make you set up the "root" password of the database server itself. Right after that, run this command at the prompt:

```
#> mysql -u root -p
```

Using the MariaDB root password defined with the `mysql_secure_installation` script, and (replacing `piwigo` and `linuxtest` with your preferred values) create a MariaDB database and user just for Piwigo by typing the five commands shown in Listing 1 in the MariaDB shell.

What these commands do is create a database called `piwigodb`, and a user named `piwigoadmin`

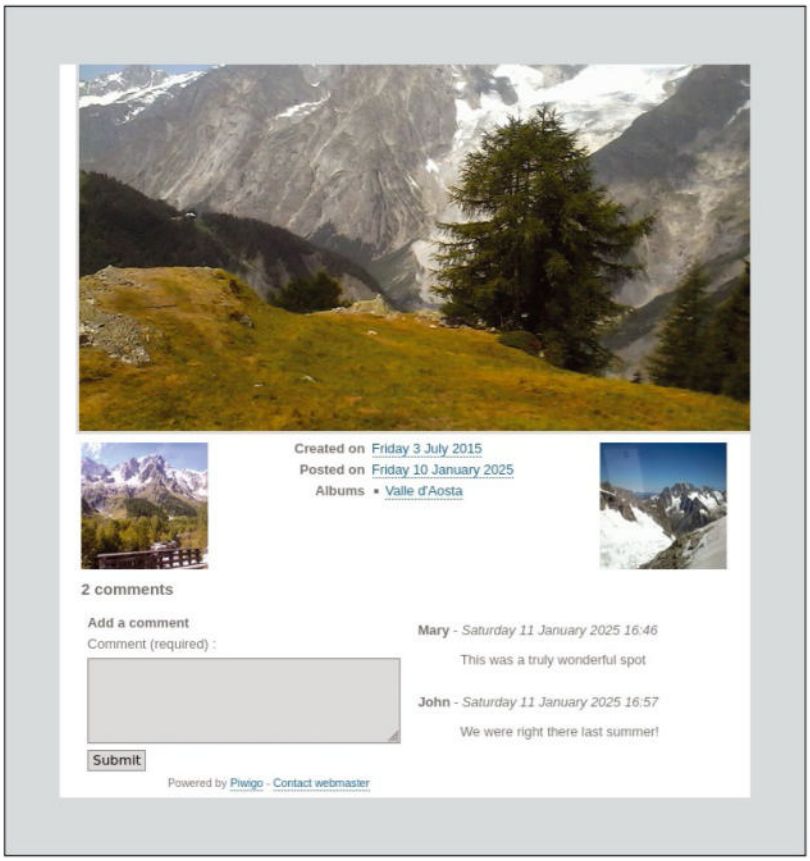


Figure 2: Piwigo users can add context to each photograph, which makes the manager even more valuable.

Listing 1: SQL Commands to Create a Piwigo Database

```
> CREATE DATABASE piwigodb;
> CREATE USER 'piwigoadmin'@'localhost' IDENTIFIED BY 'linuxtest';
> GRANT ALL ON piwigodb.* TO 'piwigoadmin'@'localhost' IDENTIFIED BY
    'linuxtest' WITH GRANT OPTION;
> FLUSH PRIVILEGES;
> EXIT;
```

with password `linuxtest`, with full administrative access to that database and only that.

Once the MariaDB configuration is over, the Piwigo documentation recommends changing these settings in the PHP configuration file for Apache, which is `/etc/php/8.1/apache2/php.ini` on Ubuntu 24 LTS and similar subfolders of `/etc/php` on other distributions:

```
memory_limit = 256M
upload_max_filesize = 100M
```

The first parameter is the minimum amount of RAM Piwigo needs to work, and the second defines the maximum size of the photographs you may upload (to avoid warnings, you may want to set the similar parameter called `post_max_size` to an identical or bigger value).

Finally, you must create an Apache configuration file in the folder `/etc/apache2/sites-available`. Listing 2 shows the file I used (called `piwigo.conf`) for the test Piwigo installation shown here, which I run on my desktop.

In a nutshell, the configuration file shown in Listing 2 tells Apache that the website called `piwigo.example.com` on TCP port 80 (the default value for web servers) is in the folder `/var/www/html/piwigo`, as well as its access privileges and log files.

Please note that the actual values of TCP port, `ServerName` and `DocumentRoot` for your installation, as well as the name of the configuration file itself, won't depend on Piwigo as such but only on how you will want to make it accessible – on its own subdomain (e.g., *mypictures.example.com*) or subfolder (e.g., *www.example.com/mypictures*). For details, please consult the Apache documentation or your web hosting provider.

Last but not least, if you run Piwigo on your own web server, you will need to run the three commands shown in Listing 3 to make Apache load the configuration file, enable the Apache rewrite module that's needed for Piwigo internal URLs to work, and then restart.

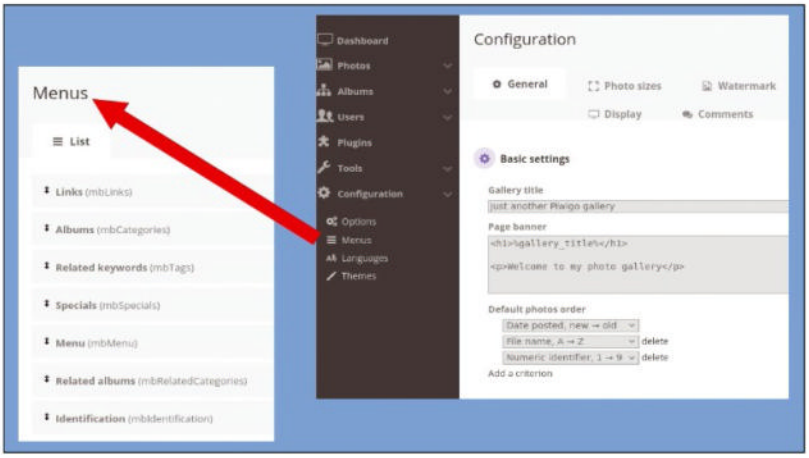


Figure 3: Three of the first things a Piwigo webmaster must configure: the website title, banner, and user menus.

At this point, you will finally be able to load your Piwigo installation's home page. For a local installation with the configuration shown in Listing 1 IT would be *http://localhost/piwigo/*. That page will prompt you to enter the database parameters, the default language, and a name and password for the Piwigo administrator account. It is highly recommended, even on single-user installations, to use that account only for administrative purposes and immediately set up another one with normal user privileges for actually using Piwigo.

Configuring Piwigo

Logging in as the Piwigo administrator, you will see the interface shown in Figure 3, which is divided into the seven sections listed in the left-side menu.

Clicking on *Configuration* and then on *Menus*, for example, opens the simple window on the left side of Figure 3, in which you can select which menus normal users see attached to each photograph and in which order.

The first place you should work in to configure Piwigo, however, is the *General* tab of the Options window (right side of Figure 3), where you must set the name of your photo gallery, its welcome message and other basic behavior, such as the default photo order or whether user self-registration should be allowed.

Other Options window tabs serve to define how many sizes of each photograph Piwigo should generate, a watermark to put on them, and how users can comment. The *Display* tab, instead, is the place to enable or disable the data, icons, and buttons that users have available to, for example, change

Listing 2: Sample Apache Configuration File for Piwigo Website

```
<VirtualHost *:80>
ServerAdmin admin@example.com
DocumentRoot /var/www/html/piwigo
ServerName mypictures.example.com

<Directory /var/www/html/piwigo/>
Options +FollowSymlinks
AllowOverride All
Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Listing 3: Reloading the Apache Configuration

```
#> a2ensite piwigo.conf
#> a2enmod rewrite
#> systemctl restart apache2
```

the sort order of photos, start a slideshow, search by tags, or see the photos inside a calendar.

Almost all the functions in the other sections of the Piwigo administrative interface are well laid out and more or less self-explanatory, so I will just mention them briefly before focusing on a few features that deserve more coverage.

Language does just what its name says: Lists all the languages in which Piwigo's interface may be localized. Clicking on *Themes* opens a directory of all the themes you can download from the Piwigo website, with buttons to activate them and, when possible, tweak their appearance.

Plugins works in the same way: Clicking on *Add a new plugin* lists all the plugins you can install on your Piwigo instance, from contact forms to creators of photo contests, shopping carts to sell photos, and tools to fix timestamps.

As you can see in Figure 3, Piwigo has five more sections I haven't mentioned yet: *Dashboard*, *Tools*, *Photos*, *Albums*, and *Users*. The first two are for regular maintenance: *Dashboard* (Figure 4) gives a graphic status report of everything that's happened recently, while *Tools* (Figure 5) lets you update Piwigo (*Update*), check past activity (*History*), execute actions such as purging the Piwigo cache or repairing the database (*Maintenance*), and synchronize photo metadata (*Synchronise*).

Inside a database-backed digital photo gallery like Piwigo, photo metadata ends up in three very different places: the actual database, the directories that separate photos by date or topic, and the photo files themselves that contain metadata in formats such as Exif or IPTC. Consequently, Piwigo offers the functions shown in Figure 5 (in the right panel) to help you keep these three data stores consistent with each other.

Photos shows the most recent photos in the default view of the Piwigo Batch Manager (more on that later). From there, clicking on the pencil icon in the top-left corner of a photo thumbnail gives what you see in Figure 6: a form where you can check and change the title, author, creation date, description, and associated keywords of the selected photo. You can also link it to as many albums as desired, which is another very powerful feature of Piwigo I will return to in a moment.

In the *Keyword* field, which supports autocompletion, you can type any keyword you want to associate with a photo. If you use this feature, Piwigo will show below each photo all of its associated keywords and also generate a word cloud of all the existing keywords, accessible from the main user menu. Of course, clicking on any keywords will show a virtual gallery of all the photos tagged in that way.

In addition to the *public* and *private* settings (mentioned later) that apply to whole albums, Piwigo also lets you categorize users as Family,

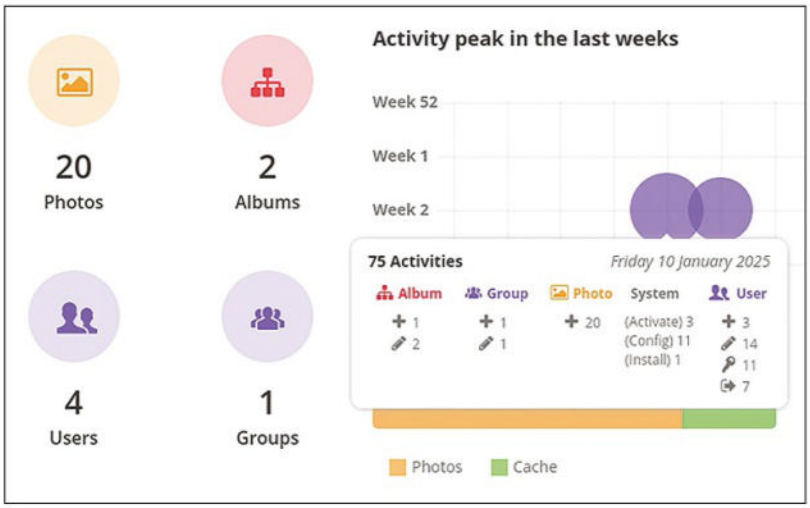


Figure 4: The Piwigo dashboard shows everything that's happening in Piwigo in a compact yet highly readable way.

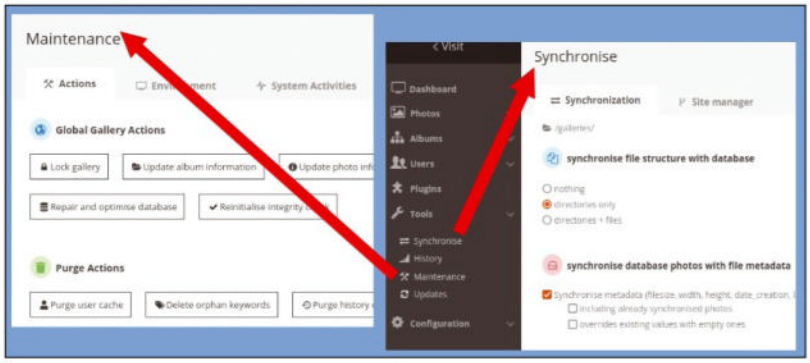


Figure 5: To make Piwigo work as expected, you must regularly clean its cache and synchronize all its data stores.

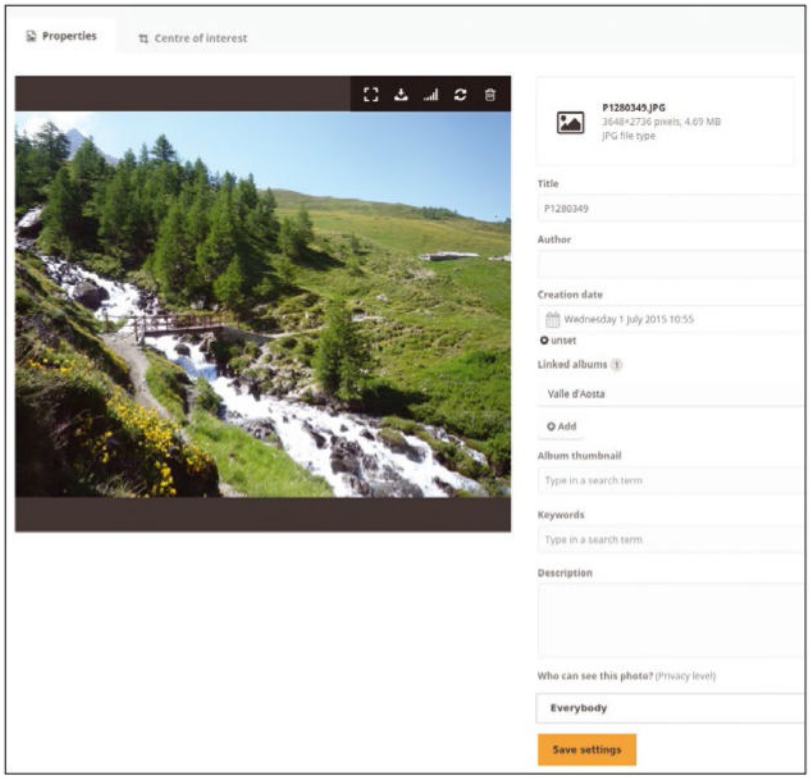


Figure 6: Each photo can have its own title, description, and keywords, as well as be included in multiple albums.

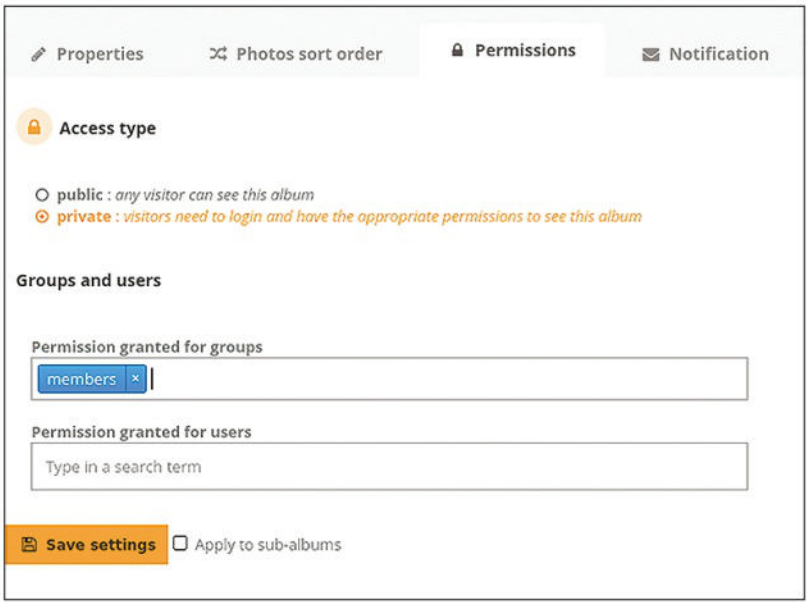


Figure 7: Just like individual photographs, each Piwigo album can have its own properties and access rights.

Friends, Contacts, or Everybody. In the last field of Figure 6, you can also specify which user categories can see the current photo.

From the *Centre of interest* tab (Figure 6), you can set the portion of the photo to be shown in the thumbnails.

Piwigo organizes photos in albums, which in Piwigo's default theme looks like Figure 1, and each album can be public or private. For public albums, everyone who knows the URL of that album will be able to browse it. If the album is set as private, only the users with an account on your Piwigo installation and the right access privileges will be able to see it. As you can see in Figure 7, it is possible to give access to specific users or to whole groups of them.

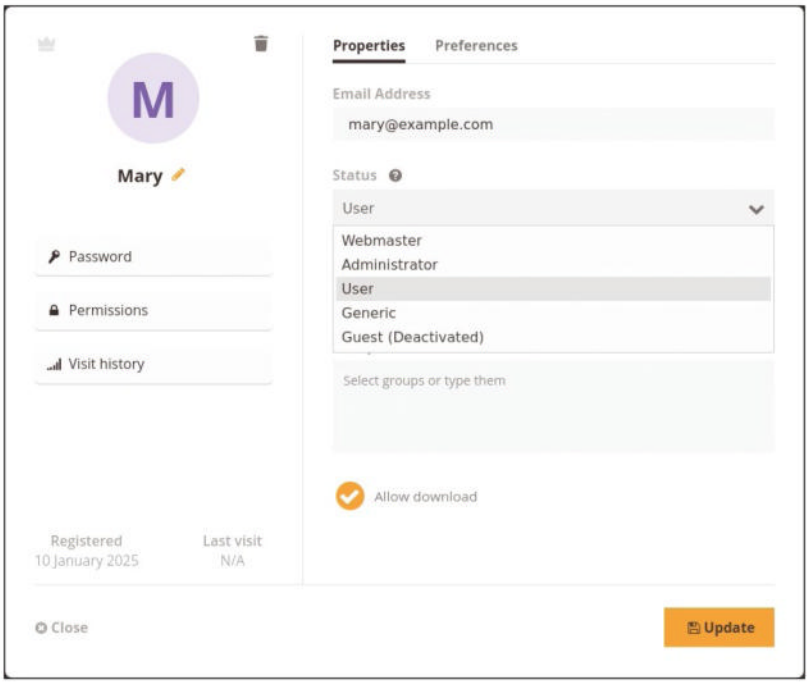


Figure 8: Five types of accounts create a clear distinction between webmaster, auxiliary administrators, and more- or less-privileged ordinary users.

In my experience, there are two more things that every Piwigo administrator must know about albums. The first, and most important by far, is that albums are not directories – not necessarily, at least. When someone uploads a photo into Piwigo, the software saves it just once in its `upload` subfolder and saves all the other sizes defined in *Configuration* | *Options* | *Photo sizes* inside the `data` subfolder. Then, the administrator and other users with the right permission can include that single copy of the photo in as many albums as desired, using the *Linked Albums* and *Add* buttons shown in Figure 6.

The other thing you should know about is comments. As I said earlier, you can enable comments for registered users and set how they work (e.g., if users can edit or delete their own comments or if they must be validated by the administrator) in the *Configuration* | *Options* | *Comments* window. However, I found out that it's not possible to actually add comments to a photo unless you also go to *Albums* | *Manage*, click on the pencil icon beside the album that includes that photo, and then click on *Authorize comments*.

Users can register by themselves if that option is enabled in *Configuration* | *Options* | *General*, but, of course, the administrator can create and configure user accounts at any time as shown in Figure 8, specifying permissions to access folders, status, email, and, above all, which groups he or she belongs to. Also note that you can also prevent a user from downloading photos and (in the *Preferences* tab of Figure 8), specify his or her preferred theme, language and number of photos per page.

Batch Manager

By now you know there is a lot you can do with photos: set their titles, date, keywords, link (or, in Piwigo's language, "associate") them to multiple albums, and more. The problem is, doing all this work one photo at a time would take so much time that nobody would do it, except on very small collections.

In practice, this problem doesn't exist, thanks to Piwigo's Batch Manager, which is the section of the Photos window visible in Figure 9. Here's how it works: First, you choose, from a drop-down menu not visible in Figure 9, one or more filters that will show all and only the photos that match the corresponding criteria. For example, you may ask Piwigo to show only the photos that are duplicates or those that are less than 3MB in size and were tagged with some keyword.

After you have chosen the filter(s) and clicked on *Refresh photo set*, you can select all the photos you want to work on by clicking on them, select what to do from the *Choose an Action* menu visible

in Figure 9, and then actually apply that action to all the photos you have selected. This is an excellent way to create many virtual albums from the same set of photo files, and in general to greatly speed up the organization, tagging, and presentation of large galleries of pictures.

Export Data Plugin

There is one last feature of Piwigo that, in my opinion, everybody who needs an online photo gallery should know. I am talking of the plugin called Export Data that can save the data stored in the database as CSV spreadsheet files. Figure 10 shows how two of those spreadsheets look in LibreOffice Calc: The top spreadsheet contains the filenames, creation dates, coordinates, tags, and descriptions of all the photos, and the other spreadsheet, with the gray background, stores all the comments made on all the photos, with their authors.

By using sed, awk, Perl, and other text-processing tools available on Linux, you can easily process and reuse all the data in any way imaginable. I, for example, plan to write scripts that will combine all the photos from the charity (and all the comments they will get thanks to Piwigo) into one nice coffee table book!

Just a Preview!

Piwigo is much more customizable than the screenshots shown in this article. You can create other auxiliary administrator accounts for other people to help you with organizing albums and managing comments. You can also lock single albums to avoid any unwanted changes. At a lower level, besides all the options that are directly accessible from the administrative interface, you can manually set options by creating a custom configuration file using the one included in the Piwigo distribution as reference.

Even Piwigo's appearance is much more customizable than shown here. For people without a real computer, or the inclination to use one, there is a Piwigo app for Android devices [2]. DigiKam users can connect to a Piwigo instance to quickly upload whole photo galleries [3] [4]. To get a more complete idea of what Piwigo can do and what it looks like, check out the official "Who uses Piwigo?" page [5] and then try it! ■■■

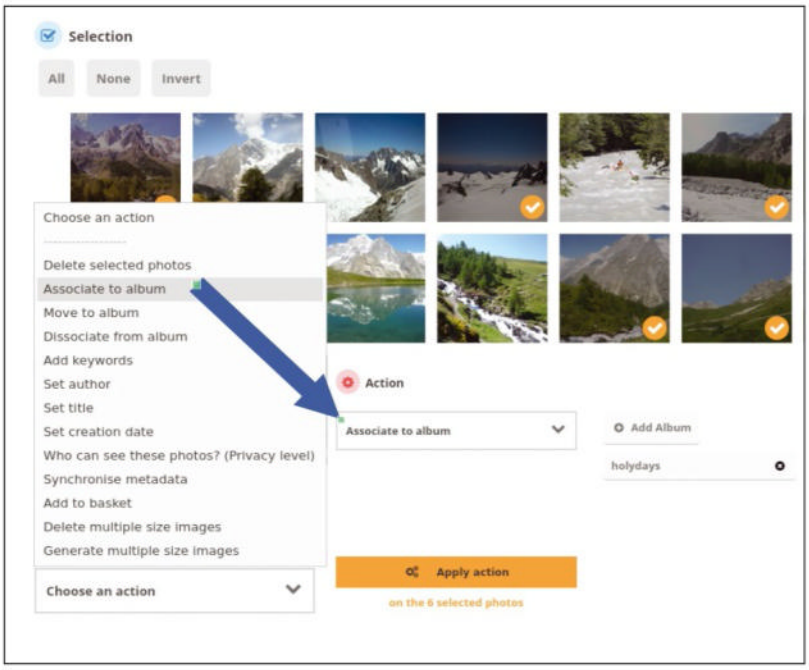
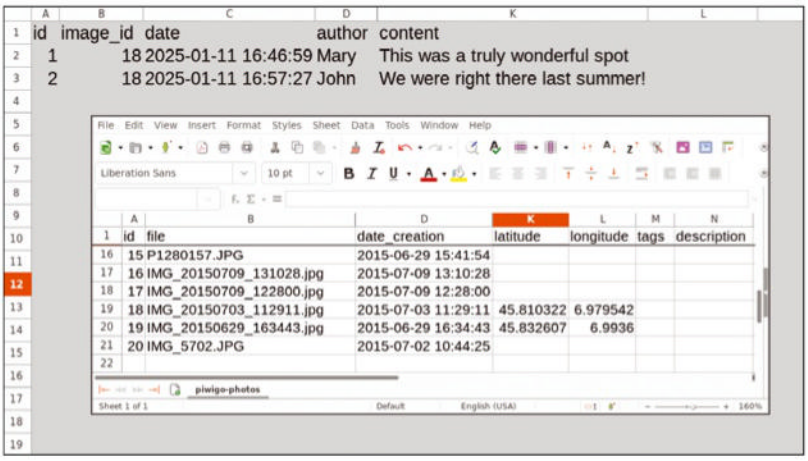


Figure 9: Processing large quantities of photos is no problem with the Piwigo Batch Manager.



id	image_id	date	author	content
1	18	2025-01-11 16:46:59	Mary	This was a truly wonderful spot
2	18	2025-01-11 16:57:27	John	We were right there last summer!

id	file	date_creation	latitude	longitude	tags	description
15	P1280157.JPG	2015-06-29 15:41:54				
16	IMG_20150709_131028.jpg	2015-07-09 13:10:28				
17	IMG_20150709_122800.jpg	2015-07-09 12:28:00				
18	IMG_20150703_112911.jpg	2015-07-03 11:29:11	45.810322	6.979542		
19	IMG_20150629_163443.jpg	2015-06-29 16:34:43	45.832607	6.9936		
20	IMG_5702.JPG	2015-07-02 10:44:25				

Figure 10: You can reuse all the metadata created by Piwigo administrators and users with the Export Data plugin.

The Author

Marco Fioretti (*https://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy, who writes about digital rights issues at *https://mfioretti.substack.com*.

Info

[1] Piwigo: *https://piwigo.org*

[2] Piwigo mobile app: *https://piwigo.org/mobile-applications*

[3] "Cloud-Based Photo Setup with digiKam and Piwigo" by Dmitri Popov, Productivity Sauce blog: *https://www.linux-magazine.com/Online/Blogs/Productivity-Sauce/Perfect-Cloud-based-Photo-Setup-with-digiKam-and-Piwigo*

[4] digikam and Piwigo: *https://piwigo.org/forum/viewtopic.php?id=31287*

[5] "Who Uses Piwigo?": *https://piwigo.org/who-uses-piwigo*

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Magazine-catalog

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.

#291/February 2025

Optimizing Linux

All the classic Linux distros are optimized for some abstract "general purpose" use case that no one matches exactly. If you want to get better performance out of your system, you'll need to tune it yourself. This month we study some steps for tweaking system and network performance.

On the DVD: Linux Mint 22 Cinnamon Edition and EndeavourOS Neo

#290/January 2025

LibreOffice Alternatives

LibreOffice is the reigning king of the Linux desktop, but some users would rather explore the other offerings before opting for the perennial default office suite. We explore some of the leading contenders.

On the DVD: Fedora 41 Live Workstation and Manjaro Xfce 24.1.1

#289/December 2024

Coding with AI

Futurists predict a day when computers will write the computer programs. Are we there already? This month we separate fact from hype to examine some of the popular AI-based coding tools and explore what they can (and can't) do well.

On the DVD: Kubuntu 24.10 and Kali Linux 2024.3

#288/November 2024

Smart Home

If you listen to megavendors like Google and Amazon, the only path to a smart home is through the cloud, but the Linux community has a better way. We'll show you some open source smart home tools with no cost and no spying.

On the DVD: Rocky Linux 9.4 and MX Linux MX-23.3

#287/October 2024

Ollama

Many Linux users are intrigued by Large Language Model (LLM) tools like ChatGPT, but if you really want to be methodical about testing and experimenting, why get tangled in the cloud? Ollama lets you run LLMs locally on a home computer.

On the DVD: Debian 12.6 and Clonezilla 3.1.3-16

#286/September 2024

Git Ready

The Git version control system is an integral part of the Linux environment. If you're looking for a better foundation in Git, or if you already know the basics and are ready to start building Git into your own custom apps, we'll make you Git ready.

On the DVD: openSUSE Leap 15.6 and Tails 6.4

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 *info@linux-magazine.com*.

FOSSASIA Summit 2025

Date: March 13-15, 2025

Location: Bangkok, Thailand

Website: *https://eventyay.com/e/4c0e0c27?code=linuxmagazine*

The FOSSASIA Summit is an annual event that has been at the forefront of open technologies for over 15 years, bringing together a global community of developers, innovators, enterprises, and educators. The 2025 edition will be a 3-day hybrid event offering in-person and online experiences

CloudFest

Date: March 17-20, 2025

Location: Europa-Park, Germany

Website: *https://registration.cloudfest.com/registration?code=CF25Linux*

CloudFest is the #1 internet infrastructure event in the world, connecting the global cloud computing industry. Form the partnerships that help you reach your business goals and have a great time doing it. Over 8,700 technology leaders will be waiting for you at Europa-Park in Rust, Germany.

DrupalCon Atlanta

Date: March 24-27, 2025

Location: Atlanta, Georgia

Website: *https://events.drupal.org/atlanta2025*

The biggest open-source event in North America, DrupalCon, is coming to Atlanta, Georgia. Whether you're new to Drupal or a long-time member of the Drupal community, you'll find new insights and connections to advance your career and your business.

Events

SCaLE 22x	March 6-9	Pasadena, California	https://www.socallinuxexpo.org/scale/22x/
FOSS Backstage	March 10-11	Berlin, Germany	https://25.foss-backstage.de/
SUSECON 25	March 10-14	Orlando, Florida	https://www.suse.com/susecon/event-details/
FOSSASIA Summit	March 13-15	Bangkok, Thailand	https://eventyay.com/e/4c0e0c27?code=linuxmagazine
CloudFest 2025	March 17-20	Europa-Park, Germany	https://www.cloudfest.com/
DrupalCon Atlanta 2025	March 24-27	Atlanta, Georgia	https://events.drupal.org/atlanta2025
KubeCon + CloudNativeCon Europe 2025	April 1-4	London, United Kingdom	https://events.linuxfoundation.org/
Linux App Summit	April 25-26	Tirana, Albania	https://linuxappsummit.org/
stackconf 2025	April 29-30	Munich, Germany	https://stackconf.eu/
PyCon US 2025	May 14-22	Pittsburgh, Pennsylvania	https://www.python.org/events/python-events/1507/
DORS/CLUC 30	May 21-23	Zagreb, Croatia	https://www.dorscluc.org/
ISC High Performance 2025	June10-13	Hamburg, Germany	https://isc-hpc.com/
OpenSouthCode 2025	June 20-21	Málaga, Spain	https://www.opensouthcode.org/conferences/open-southcode2025
Open Source Summit North America	June 23-25	Denver, Colorado	https://events.linuxfoundation.org/
Linux Security Summit North America	June 26-27	Denver, Colorado	https://events.linuxfoundation.org/
GUADEC	July 24-29	Brescia, Italy	https://events.gnome.org/event/259/
DrupalCon Vienna 2025	Oct 14-17	Vienna, Austria	https://events.drupal.org/vienna2025
CloudFest USA	Nov 5-6	Miami, Florida	https://www.cloudfest.com/usa/
NamesCon Global	Nov 5-6	Miami, Florida	https://eu1.hubs.ly/H0fRXMD0

WRITE FOR US

Linux Magazine is looking for authors to write articles on Linux and the tools of the Linux environment. We like articles on useful solutions that solve practical problems. The topic could be a desktop tool, a command-line utility, a network monitoring application, a homegrown script, or anything else with the potential to save a Linux user trouble and time. Our goal is to tell our readers stories they haven't already heard, so we're especially interested in original fixes and hacks, new tools, and useful applications that our readers might not know about. We also love articles on advanced uses for tools our readers do know about – stories that take a traditional application and put it to work in a novel or creative way.

We are currently seeking articles on the following topics for upcoming cover themes:

- Internet Privacy
- Alternative FOSS Version Control Systems (not Git)
- Cool Rasp Pi Projects

Let us know if you have ideas for articles on these themes, but keep in mind that our interests extend through the full range of Linux technical topics, including:

- Security
- Advanced Linux tuning and configuration
- Internet of Things
- Networking
- Scripting
- Artificial intelligence
- Open protocols and open standards

If you have a worthy topic that isn't on this list, try us out – we might be interested!

Please don't send us articles about products made by a company you work for, unless it is an open source tool that is freely available to everyone. Don't send us webzine-style "Top 10 Tips" articles or other superficial treatments that leave all the work to the reader. We like complete solutions, with examples and lots of details. Go deep, not wide.

Describe your idea in 1-2 paragraphs and send it to: **edit@linux-magazine.com**.

Please indicate in the subject line that your message is an article proposal.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Associate Editor
Amy Pettle

Copy Editor
Aubrey Vaughn

News Editor
Jack Wallen

MakerSpace Editor
Hans-Georg Eßer

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© starush and sdecoret, 123RF.com

Advertising
Jessica Pryor, jpryor@linuxnewmedia.com

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@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linux-magazine.com

Printed in Nuremberg, Germany by be1druckt GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950, USPS No: 347-942) is published monthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Available Starting
March 7

Issue 293 / April 2025

Trojan Horse

The best defense is to think like an attacker. Next month we focus on the methods the miscreants use to embedded malicious code in an everyday application.

Also inside:

- **Embedding Other Languages in Bash**
- **Gaming with Bazzite**
- **Secure DNS with Unbound**
- **Deep Art Effects**
- **And much more!**

Please note: Articles could change before the next issue.


Linux Magazine Preview
Issue 289 • December 2024

Get ready for another issue of Linux Magazine!

Cover Stories: Futurists predict a day when computers will write the computer programs. Are we there already? This month we separate fact from hype to examine some of the popular AI-based coding tools and explore what they can (and can't) do well.

If you are an active digital subscriber, you should have received an email with instructions to download the latest issue.

Pay less for Linux Magazine! When you buy directly from us, you get the best price and receive your issues sooner.

Order the print issue
Buy as a PDF
Subscribe to Linux Magazine

If you need assistance with a subscription, please contact:
subs@linux-magazine.com

In This Issue

Welcome: Thanks for all the Alliteration
"We should try to work with these guys. They are just getting started, but I think they might be going places, and it will be good to make the connection." These were the words of Brian, my publisher, 20 years ago, when talking about an upstart new Linux that had recently appeared on the scene.

Image © Xxxx, 123RF.com

BE THE FIRST TO SEE WHAT'S NEXT

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***