Gnome Planet - Latest News

- This Week in GNOME: #272 Shiny Icons (2026/10/30 22:08)
  Update on what happened across the GNOME project in the week from October 23 to October 30. GNOME Fellowship Sophie (she/her) says GNOME RFC Process, many fixes in glycin, CMYK images in Loupe, and more: Learn about my work in October as part of the GNOME Fellowship. GNOME Core Apps and Libraries Files ↗ Providing a simple and integrated way of managing your files and browsing your file system. Peter Eisenmann announces Many MRs have landed in Files since the 51.1 release! Here are some highlights: Jakub Steiner made some □shiny□ new placeholder icons Khalid Abu Shawarib removed blocking I/O, so Files feels snappier Khalid also added more file operation tests and fixed issues an automated audit turned up Sivaprasad Murali brought operation error messages in line with the HIG Corey Berla fixed a security issue with emptying the trash on removable devices Jordan Petridis set up automated testing on GNOME OS in GitLab CI “Programejt” made sure overly long content text in the properties dialog doesn’t get ellipsized Peter Eisenmann taught the “Open With” app search to also match executable names Software ↗ Lets you install and update applications and system extensions. Philip Withnall says Thanks to sysprof profiling by Jeff Fortin and code review by Georges, gnome-software has had some nice speedups in icon loading and downloading this week, which should make icons show up a lot faster in app listings, and reduce bandwidth and disk usage. See https://gitlab.gnome.org/GNOME/gnome-software/-/merge_requests/2466 Calendar ↗ A simple calendar application. Georges Stavracas (feaneron) announces GNOME Calendar continues to evolve with a few user facing and accessibility improvements in the past week. Thank you to everyone who submitted patches that landed this week, and everyone who reviewed and gave feedback on them! Folding Search Results Previously, searching in Calendar showed all instances of matching recurring events. Last week, we managed to merge a long-awaited patch that improves this behaviour by folding all instances of a matching recurring event into a single row! This makes searching for events cleaner and easier to parse, as the results are no longer flooded with irrelevant duplicates. See !808, by John Cardullo; reviewed by Niklas Wimmer, Alen Galinec, Zelda Ahmed, and Georges Stavracas Focus Selected Date First Continuing the long effort to make GNOME Calendar more keyboard accessible, we landed a small tweak that improves the behaviour when tabbing into the sidebar’s and event editor’s date chooser grid. Instead of focusing the first cell in the grid, focus is now properly moved to the currently selected date’s cell. This matches the expected behavior when tabbing into other grids or lists, to first focus the currently selected item. See !869, by Hari Rana; reviewed by Niklas Wimmer Improved Screen Reader Output To make Calendar more accessible, grid cells in the month view and date chooser will announce the month and day of the cell when focused. Previously, these cells only announced the day of the month, which made it very difficult for users to identify which month they were currently in! See !886 and !887, by Arak; reviewed by Hari Rana and Niklas Wimmer GLib ↗ The low-level core library that forms the basis for projects such as GTK and GNOME. Philip Withnall announces Support for XDG intents has landed in GLib, thanks to work by Ignacy Kuchciński (based on implementation work by Sebastian Wick and spec work by many others). This is the groundwork for allowing apps to register themselves as supporting opening commands in a terminal, or handling certain URI schemes. Work is ongoing to implement these specific intents. See the specification and the implementation in GLib. GNOME Circle Apps and Libraries Alexander Vanhee reports Bazaar has hit half a million downloads on Flathub! To celebrate, I’ve fixed our most upvoted issue: the Bazaar Flatpak can now install apps from user remotes. Try it

today with the beta on the GNOME Nightly remote, and report any issues you may face, especially on that niche distro only you and your grandma use. Third Party Projects Mark announces GitY 2026.7 is now out! No longer is GitY simply a git repository viewer you can now manage your repositories directly. Many features have been added this year to make it a modern version of GitG. The most recent changes include: Add tags and branches Remove tags, branches, and stashes Commit diffs show images being added, changed, or removed (if you're curious on the technical details it uses glycin) Search branches, tags, and stashes with CTRL-SHIFT-F Live repository refresh is now available as an option (default off) Can turn off the summary of lines changed per diff Download from Flahub: https://flathub.org/en/apps/com.markdeepwell.GitY Nathan Perlman reports Rewaita got an update to v1.1.8 this week! This update was mostly focused on adding new features and filling in some of the odd blanks in the theming. What's new? Added gtksourceview theme support (the editor part Gedit and Builder) Finally added hot reloading for GTK4 Fixed Firefox 157.0 theming issues Moved user guide to Github wiki Unrelated to GNOME but Rewaita now has Cinnamon desktop support Hope you all enjoy! Give Rewaita a try if you haven't, once you get sucked in you'll never want to go back. And if do you like it make sure to leave a star on Github. Installs are available of Flathub and the AUR. Rat Cornu reports Ratic got a new version 0.4.5 this week! This is not a big update, but it greatly improve performances: it is now instantaneous to reload the datas already cached during previous launches. Furthermore, scrolling in the application no longer make it lag, and the image cache size can been limited to control RAM consumption. You can check the new version on Flathub, on the repo, or even come discuss with us on our matrix room! Thanks again for all the people that contribute to the localization on weblate! Anton Isaiev reports RustConn 0.24 is out - a connection manager for SSH, RDP, VNC, SPICE, Telnet, Serial, Kubernetes, Web and Zero Trust, built with GTK4 and Libadwaita. Since 0.23: Faster SSH connects (no process spawns on the UI thread) and a Bitwarden performance pass "Set Up Key Authentication…" - generate or reuse an SSH key, install it, verify, switch over Terminal .rsc scripting + OSC 52 remote clipboard RDP clipboard direction control, QUIC proxy preset, richer MobaXterm import/export Embedded browser in the Snap (now on core26) Loopback-only tunnels and stdin-fed Bitwarden secrets Thanks to Christian Petri, Naim Shafiev and all the contributors and translators who drove this cycle. And thanks to everyone who files bugs and supports the project - this cycle was driven largely by the issues you reported. If you'd like to support development, the repo has a Sponsor link. https://github.com/totoshko88/RustConn https://flathub.org/apps/io.github.totoshko88.RustConn https://snapcraft.io/rustconn/ Daniel Elia announces Convey 51.0 has been released! Since 50.2-1 it has become a much more polished mail client; narrow windows and mobile layouts work better, notifications are clearer, with several showing at once, and spell-checking has proper dictionary support. Subject shown above the conversation Per-address signatures, with images Drag to resize images in the composer Unread filter in the search bar Unsubscribe from mailing lists Reply to List action Local Starred folder for IMAP accounts Folders in a sidebar on mobile I'm working on adding a "Favourites" section to the sidebar, so folders from any account can be shown at the top of the sidebar, and there's still bugs to be fixed, all planned for 51.1 Lastly, a big thank-you the community for getting so involved with the development of Convey, its first release to Flathub was a little over a month ago, and it's seen much more adoption than I expected! Check it out on Flathub, and check out the repo here Fractal ↗ Matrix messaging app for GNOME written in Rust. Kévin Commaille reports Only a week went by since our previous security-related update, and several upstream dependencies released even more security fixes so we bundled them into Fractal 14.3 to make sure that everything keeps running smoothly until the next release! This version is available right now on Flathub. As usual, you can also help make sure the next major version is solid by using Fractal Nightly. Help on fixing our remaining bugs is of course always appreciated! Crosswords ↗ A crossword puzzle game and creator. jrb announces GNOME Crosswords 0.3.19 is out! This version features a redesigned game layout, with smooth animations and a new bottom sheet for clues in mobile mode. In the editor, acrostics are

now out of Beta, with a much faster generator and an easier grid-building workflow. Also new: rich text in puzzle metadata, synonyms in word lists, large print templates, and tons of bug fixes. A plea for help: We received an experimental merge request to build macOS bundles. As that's outside my expertise, I'd love help testing the game and reviewing the changes. Please take a look or let me know if you can help. Get it on Flathub! (Editor) GNOME Websites Philipp Sauberzweig announces The Inter font has been installed on the GNOME WordPress instance. You can now use it on your personal GNOME blog to ensure consistency with GNOME's web typography standards. Shell Extensions Tomáš Gažovič announces RSS Feed v10 RSS Feed reached version 10, a jubilee release for the extension. The main new features: Starred articles: kept in their own group, even after they drop out of the feed or the feed is removed Configurable article actions: up to three hover buttons per article and separate actions for the left, middle and right mouse button: open, mark as read or unread, star, dismiss, mark older as read, copy link Dismiss and restore: hide articles you do not want to see and bring them all back from the header menu Article history: articles stay stored after they drop out of the feed, up to a per-feed limit Since the extension was ported to modern GNOME Shell earlier this year, it has passed 1000 downloads. Thank you to everyone who installed it, reported issues and sent feedback! Missing something in your RSS workflow? Share your idea in Feature Requests on GitHub Discussions, I read every one. Miscellaneous GNOME OS ↗ The GNOME operating system, development and testing platform Ada Magicat reports It has been a busy week with tons of fixes and 2 new contributors! Welcome John Adams and Lars Ellinger. Check out what they worked on: Lars added support for the LDAC codec to our pipewire builds. This will provide higher audio quality for bluetooth headsets that support it. !35267 John contributed 3 different fixes for Nvidia and snapd: Adapting our firmware installation for the new 615 Nvidia driver. !5122 Making Nvidia graphics cards work inside snap applications. !5146 Fixed a regression with snapd's apparmour profiles. !5123 We haven't been sitting still either: Hari, Jonas and Valentin ironed out various issues with our zswap setup. !5090, !5101, !5118, !5135 While it used to be a nightly-only feature, we think it is now rebust enough for everyone, so we enabled zswap in OS Preview as well. We now enable the extra-codecs feature by default. This was introduced earlier this year to expand codec support, but it required users to enable it manually. !5109 Jonas enabled speaker support on the Dell XPS 14 and other modern Intel laptops. !5119 Our installer now offers to erase the user's drive before installation. !5112 Valentin backported a few CUPS patches to fix printing !5009, !5097 Minor documentation touchups by Hari, zyxhere, Jonas and Tobias !4994, !5056, !5092, !5105 Events Evangelos "GeopJr" Paterakis □⚧□□ says Come visit us in FOSSCOMM, Athens! We will have a booth this weekend in the University of West Attica on 31st October and 1st November. There will be t-shirts, stickers and community members available to chat! https://2026.fosscomm.gr/ That's all for this week! See you next week, and be sure to stop by #thisweek:gnome.org with updates on your own projects!

- Jussi Pakkanen: A slightly better ThreadPoolExecutor in Pystd through types (2026/10/30 22:00)
  In the previous post we looked at a simple multithreaded task executor that behaves roughly like its Python standard library counterpart. The biggest missing feature was being able to return values from run tasks via future objects. In a fully dynamic language like Python this is fairly straightforward, whereas in a language with static typing, such as C++, things are trickier.The simplest solution is merely to make your Executor class a template, parameterized by the function you want to run. The obvious downside is code bloat. If your program has an Executor<Foo> and Executor<Bar>, all of their code is duplicated in the binary, even though 90% of it is probably the same. Also, due to the magic of template instantiation, compilation is slower. Another downside is that you can't run tasks that do different things, especially having different return value types, with one Executor.I set about trying to solve both of these issues at the same time. It turned out that if you are willing to tolerate just a little bit of OO and one extra malloc per task, one can get a fairly nice solution. As a task executor is used for things that run for a long time and

consume a lot of resources, the overhead is insignificant for most use cases. And if it absolutely, positively matters to someone, they are going to roll their own anyway. Trying to cater to those use cases would be a waste of effort in any case.These requirements point to a fairly obvious solution. The executor deals with TypeErasedFuture/Promise types that store all the task metadata via good ol' void pointers and virtual functions. The concrete Future and Promise tasks are just type safe shims on top of them. The entire implementation required the use of only two reinterpret_casts, which is quite surprising. I would have expected more, but sometimes you get lucky, I guess.The Executor gets to remain a concrete class. The only template in it is the submit method that preserves the type erased task and returns a typed future object.Using this is not quite a simple as its Python counterpart, but still fairly easy. After you have written a class that implements the executor's task interface, submitting work and consuming the results is done like this.If the return value had been something other than void, .get() would return a reference to the result object. The tradeoff between the ease of use provided by maximal templateism and minimal code generation is clearly visible in the extra boilerplate code you need to write to get things going. But in return you get both reduced binary size and faster compile times. Whether that is an improvement or a detriment is up to you.This code is not going to win any robustness prizes (though, in the modern day of AI garbage, you never know), but the end result is more than adequate for a single evening work.

- Matthew Garrett: Getting Google Cast on my Soundbar (2026/10/30 18:03)
  I have a Samsung Q800A soundbar. It ships with built-in support for the Apple Airplay protocol, but I’m not really an Apple kind of guy so that’s never been particularly helpful for me. So, instead, I have a Chromecast Audio plugged into it via TOSLINK, and that works fine - except for the fact that there’s no way to explicitly pick TOSLINK as the audio input. Instead, there’s a “Digital In” option that refers to both the optical in and eARC input from a TV. Not inherently a problem, given I’m unlikely to be watching TV while listening to music, but my TV is a Samsung Frame, a device that’s intended to display art whenever someone is standing nearby, and which infuriatingly generates an eARC signal even if it’s only in art mode and isn’t going to make any noise. This switches the soundbar away from optical and I get sad. I could plug a video cast device into the HDMI port on the soundbar and cast audio to that, but then the art would get replaced with album covers whenever I’m playing music, and again that’s not quite the ambience I want. Still, none of this is that important. Until I learned that the Australian version of the soundbar does have Google Cast support. The firmware image was downloadable from Samsung support, so I grabbed that, but unfortunately found that it was encrypted. However, if I downloaded the firmware image for the US model, the encrypted blob was identical - there’s no difference in the underlying firmware between the two of them. So, how come one has support, and the other doesn’t? I’d need an unencrypted copy of the firmware to have a real chance of figuring this out, and the easiest way to get one was to just run an ebay watch for a broken model of the same series. Once I had one in my hands I used hot air to remove the eMMC storage from the device, cleaned up the chip, and then stuck it in one of these adapters to read it on a local system. Samsung uses a weird in-house filesystem called VDFS4 on some of their devices and this was one of them, so I needed to cobble together a kernel that was compatible with their source dump for this device (not all VDFS drivers seem to be able to read all VDFS filesystems), but eventually I had it all extracted, and then I got distracted by other stuff, so let’s fast forward to this week. Samsung makes use of a modified version of the Android Debug Bridge called sdb, which is included in the Tizen SDK. The firmware showed that this device had sdbd on it, and it would be executed if a network device called “huart0” was plugged in. Samsung does, thankfully, provide the source code for their GPLed components, so I could dig through that to discover that drivers/net/usb/usbnet.c had a bunch of references to huart (which apparently stands for “HyperUART”). If CONFIG_SAMSUNG_USBNET_HYPERUART_IFNAME is set, and if the ethernet device is using either the asix or ax88179_178a drivers, and if “/opt/etc/HyperUART_IP_Addr”1 isn’t just “0”, it sets the interface name to “huart0” and in turn triggers

sdbd startup. So now I just needed to figure out how to set /opt/etc/HyperUART_IP_Addr to something else. Some digging revealed a binary called wap-factory that referenced the file, which then led to /usr/share/mobile-mgr/jsons/com.samsung.audio.waapp-verification-test.json which mentioned a “SetUartOnOff” call, which brought me here2, which led me into a bit of a rabbit hole about the Samsung WAM protocol. Long story short, you connect to either port 55001 or 56001 and make two requests over the same connection: 1 2 3 4 5 6 7 8 9 GET /connect HTTP/1.1 Host: 192.168.87.126:56001 User-Agent: SamsungMultiroom/1.0 Connection: interactive GET /UIC?cmd=%3Cname%3ESetUartOnOff%3C/name%3E%3Cp%20type%3D%22str%22%20name%3D%22uartOnOff%22%20val%3D%22on%22/%3E HTTP/1.1 Host: 192.168.87.126:56001 User-Agent: SamsungMultiroom/1.0 Connection: interactive and then plug in an asix chipset ethernet adapter into the USB port (which on mine is micro-B, so I needed an OTG adapter), and suddenly it’s sitting there on 192.168.250.250 and will speak sdb to you. sdb root on switches it into root mode, and sdb shell now gives you a root shell. I was in, and now I could investigate what was happening. First up, there was a process called “com.samsung.audio.google-cast-control-app” running, which seemed like a good start. Unfortunately it wasn’t doing anything. After some tracing I found that it was making a call to Samsung at startup time, and getting back a response telling it that Cast support wasn’t available. The request (which was helpfully dumped to stdout when I ran this by hand) included the country code, in my case “US”. Repeating the request by hand with curl and changing the code to “AU” told me it was supported, so now I needed to figure out how to get the soundbar to believe it was Australian. Some digging showed it was reading this from a config variable called db/comss/countrycode, and running 1 buxton2ctl set-string system db/comss/countrycode AU got me past that gate. Now, however, it was complaining that it couldn’t provision a key. /usr/apps/com.samsung.audio.google-cast-control-app/shared/res included a bunch of encrypted blobs and an app called teeStoreKey.elf, and tracing com.samsung.audio.google-cast-control-app showed it was trying to call that, but it seemed like it was failing with a permissions issue. Running com.samsung.audio.google-cast-control-ap directly as root triggered key provisioning successfully, and a few seconds later the rest of the Cast API came up and I could talk to port 8443 with standard Cast API calls. Unfortunately it still wasn’t working as a Cast device, though - it wasn’t advertising itself via mDNS and so nothing could see it. Querying the /setup/eureka_info endpoint gave me a clue - the ’tos_accepted’ field was false. Normally this would be set as part of initial device setup, but the Samsung app wasn’t doing that because it didn’t think the device had Cast support. It turns out that POSTing some json with {“tos_accepted”: true} to setup/set_eureka_info made it work, and suddenly there was an additional Cast device available to me. (Update: I realised I didn’t mention anything about why it was doing key provisioning in the first place! The Cast protocol doesn’t enforce cryptographic identity as such, but Google’s reference client implementation which is what almost everything uses verifies that the device is presenting an identity that chains back to Google. Unsurprisingly they don’t want the private key material to be easily extractable and used to produce clone devices, so it’s all supposed to be encrypted and only accessible to some sort of trusted execution environment, which is why keys here need to be loaded into Trustzone and never be visible in the clear to the main OS) Mission accomplished. Unfortunately the device vanished from my Smartthings home aferwards, presumably because there’s a mismatch between its reported country code and my account country. To get around that I’d somehow need to get com.samsung.audio.google-cast-control-app to either change the query without changing the global device country, or to ignore the response from Samsung (it does this every 24 hours, ugh), and that’s made more annoying by the fact that this thing will only execute signed binaries. Although, weirdly, one of the signed binaries shipped is a full Python interpreter, so if I cared enough I could probably cobble together enough Python to execute arbitrary binaries. I currently don’t. More modern soundbars don’t implement the listener on port 55001 or 56001. I don’t know how you get sdb working on them. Yes, it reads a file directly from in-kernel ↩ Good god I hate reading this kind of slop ↩

- Sophie Herold: GNOME Fellowship October 2026 (2026/10/30 15:41)
  The GNOME Foundation is supporting contributors with its fellowship program. You can help expand the fellowship program with a donation. After the release of GNOME 51 in the previous month, this month was the time were people noticed the first regressions and bugs. Hence, a lot of work went into bug fixes this month. RFC Process Landed GNOME now has a project-wide RFC process. There were a few changes that happened this month, which further simplified the process and removed the special role of teams. Since we have a consensus on a process now, I will supplement it with further documentation soon. glycin The CMYK support has been merged. You can check my last blog post for more details. Another big change this month is the move from libjxl to jxl-rs. This is the same decoder now used by Chromium and Firefox. Thanks a lot to Wonwoo for the contribution and for all the people who fixed upstream issues that the GNOME implementation revealed. Other than that, there were a lot of smaller things to address in glycin, most due to the GNOME 51 release: Fix image loading getting stuck on systems with lowered sysctl net.core.wmem_default #334 Fix libglycin’s get_mime_types blocking main thread !487 Fix orientation not parsed correctly for some images with MakerNote Exif tag #332 Fix memory growth via filling up the channel in image-rs loader for animated images #338 Fix animation threads in image-rs not terminated until the loader process exits !491 Canonicalize loader paths when reading config #340 Fix Exif decoder panic I #329 Move JPEG XL loaders into the image-rs loader while keeping the C-based editor separate !495 Fix some .cur-files being detected as TGA files #345 Fix Exif decoder panic II #346 Fix libglycin panic in construction of GlyPixelDensity with invalid data #347 Image Viewer (Loupe) Loupe benefits from the new CMYK support in glycin. Here is a comparison of nightly Loupe vs Loupe 50. Loupe with latest glycin showing the CMYK image correctly and indicating the CMYK color model under Image Format. Loupe 51 showing wrong colors for a CMYK image with ICC profile. This ICC profile oversaturates the colors to print on uncoated paper that will absorb a lot of ink. There are also some fixes that landed in Loupe this month: Set physical dimensions of generated intermediate PDF for printing correctly #417 #613 Fix fractional scaling potentially blurring image rendering (see “Snapping”) #326 Show if image uses CMYK color model !621 Other Work I reviewed Resources, which is currently incubating as a replacement GNOME Core app for System Monitor. I found a few minor issues. Other than that, the app is close to being ready for GNOME Core. A review from the Design Team is still outstanding. A topic that has been causing me a headache for a while has been how to properly implement memory limitations for glycin and thumbnailer sandboxes. There currently is no perfect solution. It looks like I will go with cgroups outside Flatpaks created via StartTransientUnit, at least for glycin. I’m happy that we finally have a decision. Support the GNOME Project The GNOME Fellowships are funded by our community. If you would like to help the GNOME project to stay sustainable, please consider donating.
- Jakub Steiner: I am playing Grid right now! (2026/10/30 00:00)
  I never liked racing "sims" like Gran Turismo. So realistic! You can bounce off the guardrails like in a pinball. Grid on the other hand was a fantastic "arcade" racer with a brilliant damage model where mistakes were punished. And the camera work! And music. Boy, that was such a treat! Nostalgia stroke me fiercely and I needed to play that game in some form. Among a sea of weird remakes, there simply wasn't a platform you could play the proper original Racedriver: Grid on, other than emulators. So I bit the bullet and dove into the craziness of RPCS3. That UX is crazy, let me tell you. In the end you get a game playable even on an old workstation, which was a surprise in itself. A much bigger surprise was the support for network play. Obviously not a single soul is playing Grid in 2026, but it's a fun thought I might convince someone to play :) 3D model: "Playstation 5 Dualsense" by AHarmlessPotato, CC-BY-4.0 Anyway, the reason to post this is to remind myself in the future how to get my PS5 DualSense controller to work on GNOME OS. As always, a udev rule. Having this config in /etc/udev/rules.d/70-dualsense.rules: # DualSense over USB (054c:0ce6) KERNEL=="hidraw*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="0ce6", MODE="0660",

TAG+="uaccess" # DualSense over Bluetooth (054c:0ce5) KERNEL=="hidraw*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="0ce5", MODE="0660", TAG+="uaccess" KERNEL=="hidraw*", KERNELS=="*054C:0CE5*", MODE="0660", TAG+="uaccess" # DualSense Edge over USB (054c:0df2) / Bluetooth (054c:0df4) KERNEL=="hidraw*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="0df2", MODE="0660", TAG+="uaccess" KERNEL=="hidraw*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="0df4", MODE="0660", TAG+="uaccess" KERNEL=="hidraw*", KERNELS=="*054C:0DF4*", MODE="0660", TAG+="uaccess" Reload the rules and replug (the ACL is re-applied on every plug, so it persists across reboots and replugs): run0 udevadm control --reload-rules && run0 udevadm trigger Check things work from inside RPCS3's sandbox: flatpak run --command=sh net.rpcs3.RPCS3 -c 'test -r /dev/hidraw9 && test -w /dev/hidraw9 && echo OK' But again, the experience of this monstrosity. Briefly looking over Alice' shoulder at what's she's cooking with Highscore, the difference in experience and polish is night & day.

- Cassidy James Blaede: Updates to the Blog (2026/10/28 00:00)
  I've been wanting to get back into more frequent blogging instead of just posting half-thought-out toots to Mastodon, turning them into a rambling thread, and then losing the thought to the depths of social media… but every time I try to write something, there are a bunch of changes I want to make to my website so I get overwhelmed and do something else. Between the recent Matrix Conference (another blog post I need to write…), travel, and taking a couple of days off of work to recover, I decided to tackle some of those changes to hopefully reduce the mental block for the future: I finally added pagination and tags, plus imported a bunch of my writing from other sites! As long as I'm here, I figured I should document the changes as a blog post, too. Pagination As my blog has grown (I have posts as from over a decade ago!), dropping every post onto a single index page has started to get a bit unwieldy. I knew Jekyll had a pagination plugin, but I hadn't set it up. I finally did that, with nicely-styled back/forward buttons. It's a little thing, but something I'd been putting off for too long. Tags With over a hundred and fifty posts, even paging through the blog is not a great experience if you're looking for something specific. I went through and tagged every blog post by topic: e.g. GNOME, Design, Flatpak & Flathub, How-To, and Web Development. Fun accident: the Star Wars tag page gets the same styling as my Star Wars pages section because of how I use path-based CSS classes for custom styling. It was a bug, but I kind of like how it's a fun easter egg, so I kept it. I also wanted a way to format the tags nicely (e.g. Star Wars for the starwars tag) and provide some more context, so I added a data file to my Jekyll site to optionally provide a display name and description for each tag. Hover the tags to see the descriptions, or check out the new tag listing to browse all of the available topics! Importing my writing Pagination and tags help a lot because now because: my blog has way more posts! I've imported my writing from elementary and Android Police so it's all accessible from one place. I wrote a ton for the elementary Tumblr, Medium publication, and blog during the decade-plus I contributed and then worked full-time on elementary; release announcements, AppCenter Spotlights, design rationale, developer tips, and more. I'd copied some of my work here in the past, but most of it was still only accessible on the elementary blog—or was lost to the depths of Medium and Tumblr. I finally spent the time to import the 80+ remaining posts I authored. Where the posts are still available on the elementary blog, I made sure to point the canonical URL there to avoid leeching any search engine juice from the original site; I just want to make it convenient to read my stuff in one place, but don't want to mess up search engine ranking or anything. You can check out the entire archive on my site under the elementary tag. Similarly (though on a much smaller scale), I contributed a few posts to Android Police before they were bought out and strip-mined by Valnet—the company described by former writers as unethical that acquires news sites and immediately lays off their staff, and threatens to sue journalists for reporting on them. Since the topics I wrote about may be of interest to readers of my blog, I added those posts as redirects (with the ability to serve them from the

site directly if they ever disappear). You can peruse those posts under the Android tag. If I find more of my writing across the Internet, I plan to pull it into this blog in a similar way; either cross-posted directly or at least indexed here with a redirect to the original. What's next? I know not to promise anything (especially timelines!) but here are some things I'd like to work on next for my blog: Per-tag styling (e.g. a unique accent color) Find some of my more verbose Mastodon threads and turn them into blog posts Automatically-generated social cards by default Search while keeping everything static Address whatever other issues I've opened Write more! If anything in this post sounded interesting, my website is open source! Check out the source code or commit history to learn how I did things. I can't promise I did everything the right way, but it works!

- Richard Hughes: SK-8855 in the year 2026 (2026/10/27 12:01)
  I've been an unapologetic fan of the Lenovo "classic" 7 row keyboard in the ThinkPad X220 and have been using the external version of it ever since. This USB laptop keyboard chassis is called the SK-8855 which was manufactured in the late 2000s and early 2010s and is now essentially unobtainium from everywhere. The SK-8855 features a TrackPoint but lacks a trackpad – which is actually a feature for me. It's rarely available on eBay as new-old stock for several hundred dollars. Over the last 10 years I've replaced the keyboard several times due to literally wearing it out, and finding genuine NMB replacement keyboards is getting harder and harder — but not impossible. For the last 10 years I've dealt with a few little quirks. Sometimes the controller board essentially sh*ts itself and crashes, but it could always be coaxed back into life by removing and reinserting the USB-A plug. It's also guilty of missing a quick left-shift press, so typing fast my sentences never get capitalized. Over the years a few people have commented on my lack of grammar skills and until very recently I had this controller board to blame. After over 15 years of being blamed for all my typos, the controller board gave up completely. This gave me the "opportunity" to replace it with a modern converter board and bring it into the 2020s. A good analogy is putting a Tesla electric drive-train in a classic Ferrari with a blown engine. This required removing some plastic supports, which made me feel like the worst computer peripheral conservator in existence. After removing the USB-C connector the new controller could be mounted and wired up to the existing USB wire. In most part it worked as a keyboard, but was set up with the en_US keymap and didn't do anything for the # key for instance. I ran the provided non-free win32 tool in Wine but that just showed some really tiny Chinese text. It also crashed a lot and I couldn't work out the pinmap format that it was writing. After being inspired by Christian, I asked Claude to analyse the updater binary and tell me what it knew, and after allowing it device access it essentially reverse engineered the hardware. I actually bought two new controller boards and so I was a little more comfortable letting Claude actually drive the keyboard itself — and every now and then it would ask me to press specific keys and tell it if certain LEDs were on or off. After a full day of Claude-ing (never leave a hardware geek alone with minimal chores for a weekend day…) I had two things: xykbms-gtk is a GTK4 standlone version of the vendor tool that works in Linux using hidraw. Using this I could swap, remap and disable keys and change all the other parameters on the controller board. I've not submitted this to Flathub as it's 80% AI generated and more than niche; if you're soldering stuff into a SK-8855 then you're more than qualified to do meson build. It does work remarkably well considering I had no kind of datasheet or firmware source code for the controller at all. The other thing was a fwupd plugin for this new keyboard controller. I actually manually created the fwupd plugin template with lots of TODOs in the right place to define the structure and then pointed Claude at the protocol documentation it had created for the GTK4 tool and asked it to fix all the TODOs in the plugin. After another couple of hours of questions and verb-ing from Claude we had a fully functional fwupd plugin that could not only update the firmware on the controller but could also write a "pinmap" that could reconfigure the keyboard between en_US and en_GB etc. I think this worked so well because we have so many self tests and linters in fwupd; Claude was forced to do the right thing before it could even run git commit. The keyboard pinmap wasn't really a "firmware" per-se, so I then also got Claude to create a WriteBlob method that allows

fwupd to write a raw payload on a CFI-type device. This will also be super useful in the future to flash SPI devices with the existing CH341a and CH347 plugins. This is only available for CFI flash devices and for users that authenticate with the admin password – for obvious reasons. I then wrote the GNOME Firmware code to be able to flash a CFI image from the GUI. It's not beautiful yet and we probably need a dedicated app, or a dedicated "mode" in gnome-firmware to make flashing CFI devices safe and easy. I'll add a screenshot when I'm happy with the UX. I'm using this keyboard right now, and maybe it can last another 15 years of abuse from me.

- Justin Wheeler: DevConf US 2026: Lessons Learned About Reaching Students in Open Source & Linux (2026/10/27 08:00)
  Walking up to the Fedora & CentOS booth on the first day of DevConf US in Boston, I was not quite sure what to expect. Although I have attended the Czech Republic variant of DevConf several times, this was my first time attending the US event. I was there as part of Red Hat's joint Fedora and CentOS community presence. What I found over those two days was not just a chance to hand out stickers and talk to Fedora Linux users. It was a useful reminder in what the Fedora event strategy needs to become. The setting: Red Hat tech on a university campus DevConf US is a community conference focused on developers and Red Hat technologies. Being held on the Boston University campus during the academic semester meant we knew we would see students. What I did not fully anticipate was how the increased student presence here would reshape what I thought our booth should accomplish. Alongside my colleagues Shaun McCance, Matthew Miller, Amy Marrich, and To Hung Sze, we staffed a joint Fedora and CentOS table. We kept the booth table light, compared to other events. We did not have any laptop demos, elaborate staging, or any fancy technical displays. We brought ourselves and our trusty Pelican event box full of stickers, lanyards, pins, and swag. Meeting the "Ubuntu is Linux" generation Most people who stopped by the table already knew Fedora, or at least recognized the logo. Most were students, but not all. We also met faculty members, industry professionals, and contributors to other open source projects. However, there was an unmistakable pattern among the students who approached us. For nearly all of them, their first real exposure to Linux happened in university computer science labs. And in university labs, Ubuntu is almost universally the default. For these students, Ubuntu was not simply a distribution preference. For them, Ubuntu was Linux. What surprised me most was that they were not defensive about it. They were genuinely curious. They wanted to understand what else was out there, how different systems fit together, and why alternative distributions mattered. The subsequent conversations with those students are some of the most interesting, fun conversations to have at a Fedora booth. Upstream at the mouth of the river That curiosity opened the door to talk about Fedora's foundational place in the broader Linux ecosystem. When you use Fedora, you are standing directly at the upstream source of enterprise Linux everywhere. For the curious, the pitch goes something like this. Every three years, Fedora branches to become the foundation for CentOS Stream, which in turn feeds directly into Red Hat Enterprise Linux (RHEL). However, the reach does not stop with Red Hat products, which is often unknown to most students. Downstream enterprise rebuilds like AlmaLinux and Rocky Linux trace their lineage back through CentOS Stream to Fedora. Furthermore, hyperscalers stand out as interesting downstream operating systems too. For example, Amazon Linux is derived from Fedora Linux. And this year, Microsoft announced that Azure Linux, their custom hypervisor container OS, will base its future architecture on Fedora. So, when you study and use Fedora, you are not just learning another distribution. You are standing at the mouth of the river. You see the innovations before they flow into every major cloud provider and enterprise datacenter in the world. The missing on-ramp The students understood that value proposition immediately. They grasped the architectural power of being upstream. However, the conversations that truly resonated were not about packaging flags or kernel versions. They were about careers. Students were asking how to build practical skills, how to level up beyond coursework, and what industry engineering teams actually look for. A handful asked specifically about how to contribute to open source and get involved with Fedora. That is where I realized what our

booth was missing. We had no clear call to action. We could explain the architecture, and we could illustrate the ecosystem on a whiteboard, but we could not smoothly shepherd someone from curiosity to contribution. We lacked a concrete on-ramp. We had no printed guide explaining the first three steps to take after leaving the venue. We had no five-minute interactive exercise someone could complete right at the table to earn something tangible. Instead, we offered a sprawling overview of "everything you can do in Fedora," which is paralyzing when you are just getting started. What hackathons taught me about student engagement Years ago, as a student at RIT and a Fedora Ambassador, I staffed more than a few Fedora booths at collegiate hackathons like HackMIT and Bitcamp. Back then, I learned a vital lesson about engaging students: they need hands-on, achievable activities. Students need something doable, self-contained, and repeatable. Most importantly, they need something they can earn. A unique reward, like an event badge or a milestone achievement, creates an enduring memory. A standard sticker gets lost in a backpack; a challenge you solved at a booth creates a personal reason to stay connected to the community. Modernizing our student event strategy Executing a commercial trade-show booth strategy at a student-heavy venue does not work. Commercial attendees often browse passively for vendor solutions; students are actively planning their futures and want to understand how distributed engineering collaboration functions in the real world. Free Software and open source are precisely that, but we have to frame them to meet their goals. If we want to translate curiosity into long-term community contributors, our playbook needs three tangible upgrades: Tailored on-ramp materials: Brief, well-structured guides that explain what our sub-projects do and provide clear first tasks for beginners. Audience-specific talking points: Equipping booth ambassadors with prompts geared toward early-career engineers, highlighting the practical career value of public upstream contributions. Interactive booth challenges: Short, five-minute exercises that allow attendees to make a small contribution, file a test ticket, or solve a puzzle right at the table. So, should Fedora & CentOS return to DevConf US? Absolutely. DevConf US is a fantastic venue, but next time we must arrive prepared for the audience in the room. The curiosity is already there. We just need to build the bridge that lets them cross over into upstream work.

- Hylke Bons: 1,447 emoji? We'll do it live (2026/10/27 00:00)
  One of the things you get to do after being laid off is a lot of reflection, but also to revisit all the old projects you used to contribute to. One of them was Pidgin: “the universal chat client”. The new Pidgin 3 app icon and a handful emoji I designed the original emoticon set for version 2 and helped redesign the user interface when the project was renamed from Gaim after a legal dispute. Meanwhile, after a huge rewrite, Pidgin 3 is making progress towards a first release and I’ve been helping out here and there with a new app icon and some status symbols for the development releases. Emoji I’ve been meaning to start live streaming my weekly app icon work to share a look in the kitchen of the icon design process, but once I gathered the courage to start a test run, the experience wasn’t amazing. I need something I can return to consistently without needing a new plan every time… So why not create a base set of 1,447 emoji for Pidgin in GNOME-style exclusively live on stream? What could go wrong? Here’s the design file to keep me accountable, so I can’t sneakily make progress offline! Twitch I’ll be chipping away at it a few times a week. Follow me on Twitch to get a notification when I go live. We’ll grab a cup of tea, put on some music, and get through the winter. If you want to support my design work for Free and Open Source projects, I’m always looking for more sponsors. Or a “sub” as the kids call it these days.
- Sam Thursfield: 25th October 2026 (2026/10/25 14:30)
  Hello, I started this blog 22 years ago now (I don’t recommend going back to the start), and for the last 5 years I wrote a monthly “status update” post, aiming to mainly write about interesting stuff I might have done in the world of open source software, which I hoped in turn would motivate

me to do more interesting stuff in open source software. The world changed a lot in that time. If I started now this would probably be a podcast with 16 listeners, instead of being a WordPress blog with 16 subscribers. I've spent a lot of time this year reading and thinking about the impact of large language models on the tech industry and open source. Its interesting because this feels like the moment, whether we like it or not, hacker culture goes mainstream. This week the world's major news outlets had to figure out what PHP is. That the BBC's cyber correspondent didn't know PHP existed has caused some joking and debate online. 24 years ago I was figuring out what PHP is. Jonathan Atkinson, who I met in a forum that was nominally about writing video games in QBASIC, asked me to make a wiki in PHP. So I did. All I knew about wikis was that, it's like a regular website except anyone can edit it. I named the project: Wiki! Here's a quote from the README: How do I install it?It's best run in its own directory. Create it one. You'll also want a directory for nodes,or you could keep them in the base wiki dir. Either way, the nodes directory has to bechmod'ed to 777. You might raise your eyebrows at that comment if you aren't yourself a teenage hacker discovering web development. Wiki! did have a login system, and the passwords were even stored as MD5 hashes rather than plain text, as I heard that was good practice. In hindsight, an attacker who knew how to set the HTTP Cookie header could impersonate anyone, like this: Cookie: wikiticket=admin. Here is a snippet of the file edit.php. Note that $node and $title are inputs you can set via a POST request, or even via URL parameters like edit.php?title="/etc/passwd": $t=$title;if ($node) { // Write new data // $file=fopen($nodedir.$t.".wiki", "w+"); $node=stripslashes(ereg_replace("(\r\n|\n|\r)","<br>", $node)); // Linebreak to <br> fputs($file, $node."\n"); Even if you just learned what PHP is today, you might see that this is not an entirely secure design. Anyway, at the time I was proud of my work, and I needed somewhere to host it. So, as you would in 2002, I registered the name "wiki" on sourceforge.net and uploaded the project code. Sourceforge, for those who haven't yet reached middle age, was the 2000's equivalent of Github. It allowed projects to have their own website, as long as they included the Sourceforge logo somewhere, and the website could use PHP. Perfect, I thought, the project's website will be a wiki! Looking back now through archive.org, I can see this code actually ran in public for many years. Here it is around launch day in 2002: https://web.archive.org/web/20021012080349/http://wiki.sourceforge.net/wiki I have a local copy from around this time, which includes indexjon.wiki: Sam, there are some serious bugs in this wiki. I keep on getting 'document contains no data' errors; get me on AIM. Welcome to grove.swoo.net. You may or may not be interested in the [weblog]. JonathanA - 29/8/02 - 11:36 am JonathanA - 29/8/02 - 07:36 am Those bugs went unfixed forever, and six months later, the front page text had been replaced with two poems, one in Italian and one in Russian: https://web.archive.org/web/20030420205758/http://wiki.sourceforge.net One is "Non chiederci la parola che squadri da ogni lato" by Eugenio Montale, written in 1923, and presumably posted by a fan. Does anyone know what it's about? A machine translation suggests it might discuss "the crisis of certainty faced by contemporary man". The other is "Anecdote for Fathers" by William Wordsworth, in Russian via an unnamed translator. Let's just sit with that for a moment. Imagine what happens in 2026 if a trivially exploitable PHP website hosted by a US corporation goes online: it'll be mining bitcoin, serving pornography and drowning in racist and antisemitic spam, probably all within the same day. In 2003, over the course of six months, my naive implementation of a wiki was gifted with… two rather thoughtful poems. Fast forward another year. I was about to turn 18 years old, was studying for A levels and mostly concerned with hanging out in the park drinking cheap cider and learning to play the guitar. I was certainly not thinking about this rather flawed chunk of PHP code that I'd written two years earlier, but via the magic of archive.org we can see that various people were genuinely trying to use it as a wiki: https://web.archive.org/web/20041121113508/http://wiki.sourceforge.net In hindsight, perhaps it attracted so much attention because I'd literally taken the subdomain: http://wiki.sourceforge.net/, unintentionally boosting the SEO reach quite a bit. The site was still active right up until 2007.

I can’t find it now, but I do remember receiving an email at some point from sourceforge.net noting that they’d taken down the wiki! website after discovering it was a major security risk. And at some point they must have (quite reasonably) reclaimed the project name as well, since http://sourceforge.net/projects/wiki no longer exists. It will always live on in my heart, though, and for the moment at: https://web.archive.org/web/20021014172056/http://sourceforge.net/projects/wiki/ The world has changed significantly since I was a teenage hacker. Some of what I learned in those days is completely obsolete, such as the QBASIC programming language and VGA graphics. I am a world expert in these irrelevant topics. Other stuff that I picked up in the early 2000s has become the foundation of my career as an operating systems developer: C programming, Linux distributions and an gradual induction into open source desktop world via Planet GNOME. And good old PHP, of course. Who knew that, a quarter way through the 21st century, PHP would be front page news? Age 18, I didn’t tell anyone in the real world that I’d written a wiki because it’d be the first time they’d heard the word “wiki”. The word evoked exotic and mysterious feelings of Hawaii in 2002. Now it universally evokes Wikipedia. People are learning about password hygiene and PHP. The arrival of AI on the scene, to me, fits into this bigger picture that began in the last century: we, the Computer People, are becoming Mainstream.

- This Week in GNOME: #271 Good News (2026/10/23 19:59)
  Update on what happened across the GNOME project in the week from October 16 to October 23. GNOME Circle Apps and Libraries Jan Lukas announces Newsflash 6.0 beta is available Lots of internal changes to clean up caches, improve performance, lower peak memory consumption, avoid crashes. If you’re looking for user visible changes there is Pdf Export using Typst A new article list layout with larger thumbnails (the old layout is still available) A re-designed article view menu (thanks Tobias Bernard) Much improved Gtk HTML view Its now possible to continue listening to a podcast while browsing other content Lots of other goodies in the complete changelog. Check it out on the flathub beta channel. GNOME Core Apps and Libraries GTK ↗ Cross-platform widget toolkit for creating graphical user interfaces. Felix reports This week, my first contributions to GTK were merged! GTK now implements the applicationShouldHandleReopen:hasVisibleWindows: delegate method on macOS. This method gets called when an app that is already running gets reactivated, for example when the user clicks its dock icon again, double-clicks the app in Finder, or clicks on the media controls in control center. This is important for apps that keep running after their window has been closed. A typical use case is a music player that continues playback in the background. Before this, reactivating such an app gave it no chance to respond, so there was no way to bring the window back. Now, the GtkApplication now receives the activate signal, just as it does on GNOME when an already running application gets launched again. Apps that already handle activate by presenting or recreating their main window will get the expected behavior on macOS now as well without any platform-specific code. The second contribution is a small fix, which solves an issue that the native “traffic lights” controls (close, minimize and zoom) stayed visible, even when the window had asked for no window buttons to be shown. Maybe some of you can guess which app needed these fixes :) Web ↗ Web browser for the GNOME desktop. adrian announces A long-standing bug that caused the “unresponsive page” dialog to reappear forever when the page became responsive again while the dialog was shown has been fixed. As an additional refinement, the dialog will be closed automatically if WebKit reports that the page starts responding again. Third Party Projects Fractal ↗ Matrix messaging app for GNOME written in Rust. Kévin Commaille reports Several upstream dependencies released security fixes so we bundled them into Fractal 14.2 to make sure that everything runs smoothly until the next release! This version is available right now on Flathub. As usual, you can also help make sure the next major version is solid by using Fractal Nightly. Help on fixing our remaining bugs is of course always appreciated! Solitaire ↗ Play Patience Games Will Warner reports Solitaire 51.1 is out! Here is what’s new: Added the Hungarian translation Updated translations: Slovenian, Ukrainian, Kazakh, Swedish Made the game centered vertically Made double-

pressing move cards to the waste in Tri-Peaks Added a deletion button for unfinished games Made the card placeholder darker in the light theme Jan-Michael Brummer announces Saldo 0.9.0 is now available. Saldo is a FinTS banking application that connects to your bank and displays your balances and transactions. This release modernises the application’s user interface with current libadwaita widgets and redesigns the account creation, transfer, category and settings dialogs for a more consistent experience. It also introduces an option to reset the database, updates the FinTS bank database, and resolves several issues affecting background operation, password changes and account refreshing. Available at Flathub: https://flathub.org/de/apps/org.tabos.saldo dabrain34 reports New Release Announcement: GstPipelineStudio v0.7.1 GstPipelineStudio version 0.7.1 is out! This bugfix release refreshes the application icon, now drawn in the GStreamer colors, and makes pipeline import keep the links to pads created at runtime. What’s new in 0.7.1: Application icon redrawn in the GStreamer red, green and blue, with the node graph running right to left Importing a gst-launch description keeps links to pads that only appear at runtime (urisourcebin, demuxer outputs) Big Buck Bunny preset switched to the AV1/Opus sample Highlights of the 0.7.x series: New application icon Nodes colored by type: source, transform, sink Swap a node for an alternative element from its context menu Tags tab listing tags from bus messages Sortable columns in the element and pipeline lists Suggestions reuse a compatible free element already in the graph Space bar toggles play/pause, s stops the pipeline Explicit properties are kept when importing a pipeline German translation added Participants: Stéphane Cerveau — maintainer and main contributor Translations contributed through Weblate Upgrade Now! To get the latest version of GstPipelineStudio, visit the project’s page and check out the Changelog for more details. Happy streaming! Shell Extensions storageb announces Create your own Quick Settings toggle buttons! Custom Command Toggle lets you add fully customizable toggle buttons to the Quick Settings menu. Turn shell commands, services, or your own scripts into toggles integrated directly into the GNOME panel. Key Features: Run commands or launch scripts from toggle buttons Configure initial toggle states and startup behavior Create custom dropdown menus from toggle buttons Synchronize toggle states with a command’s output Assign keyboard shortcuts to buttons Import and export button configurations The latest release adds support for creating dropdown menus on toggle buttons to run additional commands, adding even more functionality to your toggling experience. Italian, Dutch, and French translations have been added thanks to some contributions from the community. Other improvements include some under the hood bug fixes and performance improvements. The extension is available on GNOME Extensions. For more information, see the documentation. Events Daniel Galleguillos Cruz says GNOME Latam 2026: Call for Participation We’re excited to announce GNOME Latam 2026, the regional gathering of the GNOME community in Latin America! Dates: November 12–14, 2026 Venue: Universidad Anáhuac Mayab, Mérida, Yucatán, Mexico Format: Hybrid (in-person sessions with online streaming and remote participation) Event page & registration: https://events.gnome.org/e/latam2026 About the Event GNOME Latam 2026 brings together developers, system administrators, educators, designers, and open source enthusiasts from across Latin America. Building on previous regional collaborations, including our work with CCOSS, the event is more than a conference: it’s a community effort to grow, empower, and sustain the GNOME ecosystem in the region. The program will include keynotes, technical workshops, panel discussions, and networking sessions, both in person and online. Our goal is to share knowledge and build lasting collaboration across borders. Our Mission To drive the adoption, development, and sustainability of GNOME and its software ecosystem in Latin America, while promoting the core values of free software: freedom, collaboration, openness, and transparency. We believe access to technology should be universal, and that diverse perspectives, from experienced contributors to newcomers joining through community initiatives like CCOSS, are essential to inclusive innovation. Goals Outreach: Showcase the latest developments, roadmaps, and releases in the GNOME ecosystem. Learning: Offer sessions for everyone, from users getting started with free software to developers who want to contribute code, design, or QA to

GNOME. Regional community: Strengthen the network of free software communities in Latin America through mentorship and collaboration with partners such as CCOSS. Accessibility and inclusion: Use a hybrid format to remove geographic barriers and welcome participants from all backgrounds. Get Involved Code of Conduct: All in-person and online activities follow the GNOME Code of Conduct to ensure a safe, respectful, and harassment-free environment for everyone. We hope to see you in Mérida or online! GNOME Latam Team latam@gnome,org That’s all for this week! See you next week, and be sure to stop by #thisweek:gnome.org with updates on your own projects!

- Flathub: Flathub Discourse is now read-only (2026/10/23 00:00)
  As of today, Flathub Discourse is read-only. Existing threads stay available, but it is no longer possible to start new topics or reply. We opened the forum back in 2020 as a place for asynchronous communication that isn't GitHub issues or IRC. Realistically, it never took off, becoming another issue tracker to keep an eye on. A forum where people post and wait for an answer that never comes is worse than no forum at all. Here's where to go instead: Help with packaging and submissions: the Flathub Matrix channel, or the Flatpak channel for questions about Flatpak itself. Problems with an app: the issue tracker of its repository, linked as "Manifest" on the app's Flathub page. Issues with Flathub itself go to flathub/flathub. App requests: Flathub doesn't take them. Ask the app's developers to submit it, or do it yourself. Runtime announcements: runtime projects such as freedesktop-sdk are welcome to post them on this blog. Thanks to everyone who answered questions there over the years.
- Jonathan Blandford: Crosswords 0.3.19 — New Layout and Acrostic Editor (2026/10/20 13:00)
  Time for a new Crosswords release! This is a good one. This past cycle features two major changes: the game got a major visual overhaul, and the Acrostic Editor is no longer in Beta. This release is dedicated to Luciana. She contributed the adorable artwork for libipuz, and is a budding cruciverbalist. She’s currently having a bit of a tough go of it healthwise, so please join me in sending her good wishes. We’re all rooting for you! New Game Layout The game mode got a complete rework this cycle. We moved the Across /Down labels into the header, and added independent header bars for each column. This work was designed by “gnoman”, who did a fabulous initial mockup, and then provided a lot of additional feedback as we iterated through possibilities. The end result looks so much better than before, and is orders of magnitude better than anything I dreamed of when I started this project. Initial mockup from “gnoman” Every aspect of the game looks better, especially single-column mode. That now has a bottom drawer for the clues. Take a look at how it resizes:
  https://blogs.gnome.org/jrb/files/2026/10/Screencast-From-2026-10-19-16-36-11.webm I had to heavily adjust the reflow animation to deal with the updated appearance. For better or worse, this layout cannot be done with stock GTK/libadwaita containers and needs a complex custom layout widget. It’s a pain to edit, but it also means that we control every stage of the animation process. We handle things from reparenting buttons to panel changes. The direction switching even cross-fades between the label and the toggle instead of snapping. It’s a small thing, but it makes the whole animation silky smooth. “Gnoman” also contributed animated SVGs for the “How to Play” dialog. Florian migrated us to the new GTK SVG animation framework, and we’ve ended up with an even slicker, charming intro screen. Acrostic Editor Is No Longer Beta A few years ago, Tanmay added acrostic support to the game. This puzzle type was fun-looking! But since we didn’t have a ready source of them, he started working on editor so we could write our own. He got something basic working but it wasn’t reliably usable. The problem was the grid autofill feature: it was just too slow. It worked by doing a DFS through the possible anagrams with a random answer shape. Sometimes it found results, most of the time it did not. This cycle, I went back to try and fix it and now have something usable. As sometimes is the case, the trick to get it working turned out to be really simple: We just had to try a new random shape after 30K iterations or so. It was a small change to the code

— just a counter. But we ended up with something that could find grids with regularity. Once that started working I went back and fixed up the rest of the workflow. The Quote/Source entries are now instant apply and you can manually edit entries in the list. There's an info panel that gives acrostic stats and an overlay pane to mark progress. It looks a lot better — and works! I've used it to create three acrostics already and have more coming. Take a look: https://blogs.gnome.org/jrb/files/2026/10/Screencast-From-2026-10-19-18-14-20.webm New Generic Widgets As part of working on the Acrostic Editor, I wrote some new widgets. These are not crosswords-specific, and are written so that people can cut-and-paste them into their app (if it's GPLv3 compatible). They're targeted for internal use but maybe would be useful for others. The more interesting of the two is XwdRichTextRow. It's an alternative for AdwEntryRow that uses a GtkTextView instead of a GtkEntry. It has two features that the libadwaita widget doesn't: An :n-rows property to set the number of visible rows. If set to 1, it matches the pixel extents of AdwEntryRow. But larger numbers will provide additional lines of editable space. A vertical scrollbar is provided once the row count gets large enough. Rich text editing. It supports bold, italic, strikethrough etc, and provides a convertor from pango markup to-and-from the widget. I wrote it in such a way that a markdown convertor could be written as well. It's not as comprehensive as AdwEntryRow yet: it doesn't support prefix/suffix widgets, emojis, or an apply button, and there are probably a lot of bugs. It also will never support GtkEditable, due to the GtkTextBuffer underpinnings. But it may be useful and fill a niche that I, at least, have felt a void for a while.
https://blogs.gnome.org/jrb/files/2026/10/Screencast-From-2026-10-19-16-05-18.webm I also wrote XwdInfoButton, which is basically a simple menu button that creates explanatory popovers. It should be used sparingly, but for a really complex app like the editor I think it can help demystify some sections. It takes a markdown string as its format. Info Popover Let me know if you have any interest in using these widgets. Non-Distro Builds, And a Plea For Help Crosswords has built on macOS for a while thanks to Vinny's heroics, but nothing stopped us from breaking it. He's fixed that: we now have a macOS arm64 CI job, building against Homebrew with Apple clang. Now macOS-specific errors can be caught at commit time. Vinny also provided an MR for building bundles as part of the build process! I'd love to have bundles available to go with the flatpaks we create. However, I have neither a Mac to test on, nor the skills to evaluate this MR. I would deeply appreciate if anyone can help test and evaluate this! Crosswords running on a Mac Florian also did another build of Crosswords for Android and created another APK. It's still pretty buggy, but it's significantly better than the previous release. I think we can get it usable with a little more effort. Android Build Other Features A grab bag of the rest: The metadata tab now has rich text editing throughout and was reorganized into a vertical layout. The Language and Country rows let you set the puzzle's $LANG and Charset. A new layer warns about letters that aren't in the alphabet. It was shockingly complex to write. Rows for setting $LANG Word lists now show synonyms in clue definitions. Definition and synonyms of "RAISE" The Editor preview window uses the real game widget. We enabled large text for printing. Toluwaleke did most of the work already a few releases back, it just needed a template. Added support for the Czech CH digraph, mirroring the existing Dutch IJ digraph. Laureen has been steadily working on a grid generator for vocab puzzles. None of it is visible yet — it's all generator code — but it's a solid basis for the next game type. I fixed a massive leak with all the PuzzleTasks. Turns out we weren't freeing them at all due to a ref loop. Not a big deal with the normal autofill behavior, but once we started running it every keypress the leak grew fast. Thanks! This release, thanks to: Rosanna for tireless help with UI word-smithing and editing, as well as endless patience. "Gnoman" for the amazing design for the revamped play mode, the animated SVGs, and general design support. Tobias for design support and advice throughout the Circle process. Florian for the port to GTK4's new SVG widget and other fixes. Vinny for macOS input method support and the macOS arm64 CI job, and in general amazing build gardening. Michel for build and license advice. Laureen for continued work on the vocab editor. Federico for general advice. Alice for general advice, and for fixing my issues with libadwaita.

Eleanor, Sophia, Michael, and Rosanna for puzzle play-testing (coming soon!) Peace

- Sebastian Wick: Varlink observability (2026/10/18 13:37)
  A few days ago systemd gained tools which allow you to monitor Varlink traffic. A new Varlink service (io.systemd.VarlinkMonitor) allows you to set up a ring buffer which will get filled with packets containing the traffic which matches the filters you provide. This provides the functionality that org.freedesktop.DBus.Monitoring provides for D-Bus. A new varlinkctl monitor verb allows you to use this feature in terminals and scripts. This provides the functionality that dbus-monitor provides for D-Bus and uses io.systemd.VarlinkMonitor under the hood. If you pass --json short to varlinkctl monitor, it produces JSON-seq which can be passed into jq --seq to further process the captured data. I presented this in the broader context of observability with Varlink at All Systems Go. Slides and video of the talk are available online. This is a good start, but I think we should add more tooling around it. Especially Bustle would be an interesting target because it would allow us to capture D-Bus and Varlink at the same time and get a global view of what is going on in our systems again.
- Jakub Steiner: Liberec Dam (2026/10/17 00:00)
  Liberec Dam 2026 Watch on YouTube There's only a few places I get to take visiting friends to in Liberec. Sadly at the moment almost all of them are out of service. The city dam, Harcovská přehrada, is one of them. A fraction of the size of a neighboring Mšeno, it's set close to the middle of the city and is a wonderful walk/running place, despite only half having a foot path. In the autumn of 2022, a full reconstruction finally began, a mere century after the dam was built. Quite a massive operation, with 26,000 cubic meters of silt dug out, the original masonry injected with a fresh waterproof jacket from a brand new gallery dug right under the dam, and the bottom outlets were swapped for beefier ones. Being a listed landmark, the dam got to keep its pretty face while the guts were replaced. A three-year job that -- in the finest local tradition -- took a bit longer and cost north of EUR 20 million. It's worth noting that Liberec is one if not the most indebted cities in Czechia. In the meantime the city had a moonscape where its living room used to be. I went to investigate the drained bowl like everyone else and came back with a portrait short of the eerie emptiness (mixed with some shots from the wider neighborhood). Empty Dam Watch on YouTube This year the water finally came back. Not yet finished, the lake is once again the city's front yard, has a refurbished beach, and a dam that's now rated to pass a 10,000-year flood. A figure I sincerely hope remains purely theoretical. To celebrate, I did the only reasonable thing: sent up a flying camera and cut the results to a tune. Sadly the dam claimed another one of my birds (currently the score is 2:0). A mistake had cost me one of older 5" as a broken OSD, ancient battery and water can lead to brisk ends. Luckily the tandem flight with a friend was done on my 4" and I have a video to share above. The soundtrack was composed, as is the tradition around here, on the Dirtywave M8. Should watching atmospheric drum & bass slowly materialize out of a tracker over the course of a couple of hours sound like your idea of an evening, the recorded stream is up on YouTube.
- This Week in GNOME: #270 Visualized Community (2026/10/16 21:37)
  Update on what happened across the GNOME project in the week October 9 to October 16. Miscellaneous balooii reports After the initial summer release of Contributor Atlas I can now announce the next version and additional dataset. This release now supports multi-project communities, a lot of polish and a few visualization which are either fun or insightful. Hopefully both. The new dataset is the GNOME community with many of its projects and their contributors. Have fun exploring over at https://balooii.pages.gitlab.gnome.org/contributor-atlas/ GNOME OS ↗ The GNOME operating system, development and testing platform Ada Magicat announces We are excited to announce GNOME OS Preview! This new edition of GNOME OS is built from the latest stable release of GNOME, unlike our nightly edition. Targetted at early adopters, it should provide a much more reliable experience. More details on Adrian’s blog: https://blogs.gnome.org/adrianvovk/2026/10/16/gnome-os-preview/ Downloads at the

usual place: https://os.gnome.org GNOME Core Apps and Libraries Libadwaita ↗ Building blocks for modern GNOME apps using GTK4. Alice (she/her) □□□ announces AdwActionRow now has the :body property, allowing to add single full width child, like a slider, illustration or a toggle group Calendar ↗ A simple calendar application. Zelda Ahmed □□ □□□□ reports GNOME Calendar Highlights of the Week This week we had quite a few nice improvements land to Calendar this week, including a lot of clean-up and back-end improvements, and a few user facing changes. Thank you to all the amazing community members for their efforts to make Calendar even better this week! Decrementing and Incrementing Years in the Month Spin Button In the month spin button, going to the previous month from January will now jump to December of the previous year. Similarly, going to the next month from December will jump to January next year. This had been a long requested feature for quite a while. It seemed easy to implement, but it ended up being more difficult than desired. Since Calendar has several views, many of the widgets' properties are shared. When one of the widgets changes, every widget updates accordingly, which made it difficult to have the years and months changing at the same time. This required building on top of the optimisations made to the date chooser that landed a few weeks ago. See !853, by Hari Rana Zoom Shortcuts To make it easier to control Calendar using just keyboard shortcuts, we added the ability to use Ctrl+/Ctrl- to zoom in and out of the week view. This joins the ability to zoom in and out the week view using a trackpad or holding down Ctrl and scrolling the mouse wheel. See !859, by Arak Configurable Month Rows The month view in Calendar is split into rows, with each row representing a week in a given month. The number of rows in the month view was hardcoded to 5 to span all the weeks in one month. While this works for most people, for users with non-standard display sizes or busier schedules, a hardcoded 5 rows was too limiting. The first steps to make the month view more flexible landed this week, with the ability to change the number of rows displayed in the month view. This feature is still experimental, currently requiring the user to change the value through an internal setting using a program like dconf editor, but exposing this option in the user interface is planned for later this cycle. See !856, by Georges Stavracas Third Party Projects xjuan announces Cambalache 1.2 Released! Hello, I am pleased to announce a new stable release. Cambalache is a WYSIWYG (What You See Is What You Get) tool that allows you to create and edit user interfaces for @GTK 4 and 3 applications. This new version depends on @gnome SDK 51 and includes all new properties and types from Gtk and Adwaita! It also removes the file hash from the project file which should prevent unnecessary git conflicts. https://blogs.gnome.org/xjuan/2026/10/14/cambalache-1-2-released/ Jan-Willem reports This week I released Haystack, a new GNOME app for editing OpenQA needles. OpenQA is an automated test tool for operating systems, and it uses needles (annotated screenshots) to verify the output. Haystack makes it easy and convenient to create and maintain needles. Get it on Flathub Haydn Trowell reports Typesetter, the Typst editor for focused writing, gets its biggest update yet with version 0.16. The headline feature is sandbox-compatible multi-file projects. Typesetter now uses a new default archive-based project format, .typz, that keeps your document and everything it references (images, imports, and other assets) together in a single file. Projects work the same wherever you open them, and they're easy to share and back up. Because Typesetter now only ever needs access to the files you explicitly open, never entire folders, multi-file Typst projects now work easily and reliably within the Flatpak sandbox. A project can hold as many sub-files as you like, so you can split a long document into chapters or sections, keep reusable functions and styles in their own files, and bring everything together with #include and #import statements. This replaces the previous "project folder" workflow, which was required to make imported files available to single-file .typ documents. That approach presupposed access to whole directories, which the sandbox is designed to prevent, and it was easy to get wrong. Removing it means there is now one secure, consistent way to include files in a document. Typesetter can still open and edit standalone .typ documents, but they can no longer reference external files. To include assets in an existing document, open it and choose "Save As Project" when prompted, then add the files through the

navigator. Other noteworthy features include: GUI bibliography manager: You can now add and edit references through a dedicated graphical interface. Adjustable preview pane: The preview pane can now be manually resized to suit your workflow. Comments navigator: A new navigator lets you list, view, and jump directly to author comments throughout your document. New templates: Typesetter now includes several additional templates to help you get started with new documents. (More to come.) Since the previous release announcement, Typesetter has also gained advanced export to HTML and Markdown, making it easier to share documents in other formats. There have also been a range of performance improvements, bug fixes, and smaller feature updates across the application, continuing Typesetter's focus on being a fast, distraction-free writing environment. Typesetter 0.16.0 is available now on Flathub. Riyan announces Kaori an open-source comic reader for GNOME, which allows you to discover, download and read comics with a beautiful libadwaita UI and natural touchpad gestures. The name Kaori is a small tribute to Kaori Miyazono from Your Lie in April, a character who left a lasting impression on me. Kaori is now available on Flathub Haydn Trowell reports A new update to Kotoba, the Japanese–English dictionary app, is live on Flathub. This release includes some code cleanup, internal optimizations, and an upgrade to the GNOME 51 runtime. Grab it here: https://flathub.org/apps/net.trowell.kotoba moosee says Moose — A native GTK and libadwaita app for local and cloud AI chats Moose 0.6.0 is out! I started Moose to give local AI a comfortable home on the Linux desktop. This release brings cloud models into the app while making it clearer what you share and where it goes. Connect to Ollama Cloud, Groq, OpenAI, Anthropic, or Google Gemini with your own API key, stored in your desktop keyring. When you want to keep inference on your device, Local Only restricts it to Ollama managed by Moose. For remote providers, choose whether Moose asks before sharing messages in each chat or remembers your choice across chats with that provider. Files have a separate permission. You can review the context before it is sent and reset permissions in Privacy settings. I'm especially happy with the new Files view. It gives attachments and library documents a place of their own: search and filter them, preview a file, find its original folder, return to a related chat, or attach it to the current conversation. There's also a broader local model catalog for chat, vision, reasoning, and coding. Waiting and reasoning indicators are clearer, and remote Ollama connections now require HTTPS. Conversations, drafts, and the document library stay on your device. Content you choose to share with a remote provider is handled under that provider's data policy. If you give Moose a try, I'd love to hear what you think. Get it on Flathub: https://flathub.org/en/apps/io.github.moooossee.Moose Source code: https://github.com/moooossee/moose Project website: https://moooossee.github.io/ Release notes: https://github.com/moooossee/moose/releases/tag/v0.6.0 Lockpicker ↗ Recover passwords from their hash Sjoerd Stendahl says This week I had a design discussion with Philipp Sauberzweig about Lockpicker. Before making some bigger changes, I decided to start with the low-hanging fruit. So this release mainly focuses on minor adjustments and some bug-fixes that were already in the main branch of the code. The biggest visible changes are a hash picker that includes the custom modules, the ability to select/unselect rules by drag and drop, and the proper use of Adw Comborows throughout the application instead of the old Gtk pickers. Lockpicker is a tool to help you to recover passwords based on a hash, it's essentially a GUI front-end to hashcat. You can get it on Flathub here. Gitte ↗ A simple Git GUI for GNOME Christian says Gitte, a simple Git client for GNOME built with GTK4, libadwaita and Relm4, just got its 0.11.0 release! The headline feature this time is search in the commit graph: you can now find commits directly in the graph using the same search syntax as the commit log. Search also learned to match commit IDs and ID prefixes (short IDs), and author searches now match email addresses as well as names. And with the new sidebar search (Ctrl+K), you can quickly filter branches, tags and remote branches. Commit messages got a few new tools, too. You can add trailers using configurable templates, set defaults per repository, and sign off commits with your identity. When squashing commits, trailers are merged, and edited commit messages are preserved when a history rewrite pauses. There is also a new button in the

commit details to copy the complete message. Amending a commit is now much more flexible: you can edit all changes from the previous commit, including unstaging changes to remove them from that commit. A few more things can now be tailored to your repositories. You can choose whether they open in Changes, Current Branch or Commit Graph, either globally or per repository. Gitte also picks up a logo.png in the repository root, following the project-logo convention used by GNOME. logo.gif, logo.jpg and logo.jpeg are supported, too. On the UI side, selection handling in the working copy and stashes views is smoother, with selections restored after refreshes and filtering. The repository name now shows up when the sidebar is collapsed, preferences tabs and repository settings have been reorganised, and spacing, compact-mode padding, diff icons and layouts with long translations got some polish. And of course there is the usual pile of fixes: discarding hunks or selected lines during a rebase no longer runs into hunk-index errors, file names containing wildcard characters are handled correctly, and conflict diffs are loaded against HEAD. Status errors are now reported instead of making the working copy look empty, and diffs use consistent path prefixes regardless of your Git configuration. Failed clone attempts no longer risk deleting an existing destination folder, rebases resume correctly when commit signing is enabled, and notification messages display characters such as < and & correctly. macOS also got more reliable detection of command-line tools. Under the hood there are updated German and Ukrainian translations, fresh dependencies and bundled Git and Git LFS versions, and a move to the GNOME 51 Flatpak runtime. Get it on Flathub, for macOS or have a look at the Code. Bouncer ↗ Bouncer is an application to help you choose the correct firewall zone for wireless connections. justinrdonnelly announces Bouncer 51 is out! It may be fall here, but it was time for some spring cleaning. This release includes a bunch of small cleanups. The most siginificant user-facing change is the upgrade to GNOME 51. Get it from Flathub. I hope you enjoy it! Shell Extensions Just Perfection reports We are calling for volunteers to help the GNOME Extensions team with code reviews and Shexli linter improvements. If you are interested in joining the review team or helping with Shexli, please reach out to us in the GNOME Extensions Matrix channel. Just Perfection says A new rule has been added to the EGO review guidelines that makes the GJS shebang mandatory for all standalone GJS scripts. This rule was introduced because standalone GJS files have a different architecture than standard modules. They run the mainloop in the global scope, which should never be done by extensions in a shell process. To make it clear during code review that these files are standalone scripts rather than modules loaded into the shell process, they must start with the GJS shebang. swink announces Now Playing Card got a few more updates since last time. Beautiful music controls for the top bar: a minimal animated indicator and a compact card with cover art, a seek bar and playback controls, for any MPRIS player. After the last post, many people sent in feature requests, and a lot of them are now in. Thanks to everyone who wrote in! Recent updates bring: a new icon (grey hills in fog) and a matching spectrum shape, Waves, where the hills drift by and rise with the sound an optional “Follow all sound” mode: the spectrum reacts to any sound on the system, including calls and games, not only the player a lighter panel icon that runs on its own timer instead of the frame clock, with a frames-per-second setting (10–60) an option to show the playing app’s icon in the panel, next to the equalizer or in its place album art can be turned off, so nothing is fetched from the web equalizer bars can use the system accent colour (GNOME 47 and later) right click on the panel button can play/pause or skip instead of opening the menu wide video thumbnails fill the cover square instead of being squeezed a reset-all-settings button on the About page Source code is on GitHub. That’s all for this week! See you next week, and be sure to stop by #thisweek:gnome.org with updates on your own projects!

- Adrian Vovk: Introducing GNOME OS Preview (2026/10/16 17:47)
  It’s been two years since my initial post about making GNOME OS an end-user general purpose OS. Today, we’re excited to announce that we’ve hit a major milestone! TL;DR: We’re launching a new edition called “GNOME OS Preview”, which is built from the latest stable GNOME release.

This means that GNOME OS is no longer just a nightly OS for GNOME developers and testers! The Road to Stable Even though GNOME OS has been usable on real hardware for a few years now, its appeal was limited. We've only had the Nightly edition, which is constantly rebuilt to ship the latest git commit of every GNOME component (and a few others, like systemd and Flatpak). This form of GNOME OS is extremely useful, since GNOME developers and designers can daily-drive it to test the latest changes. However, it's not something we'd recommend to non-developers. Over the past year, we've seen a lot of adoption of GNOME OS Nightly amongst GNOME developers, especially following the success of last year's Summer of GNOME OS challenge. This has led to a significant uptick in contributions and initiatives aimed at improving user and developer experience. For instance: Thanks to Jonas Dreßler's, Valentin David's, and Sebastian Wick's zswap enablement efforts, GNOME OS now handles devices with limited amounts of RAM without OOM stalls. Jonas also found a hack that lets GNOME OS dynamically grow the usable on-disk swap space on-demand, which makes zswap perform significantly better than usual. The recently released Test Center app allows you to install CI builds of Flatpak apps from a merge request, making it super easy to test experimental changes before they are even merged. On GNOME OS Nightly, Test Center can do the same with system components by applying them as a system extension. Jordan Petridis's new Toolpak project aims to provide a clean solution for distributing and installing command-line applications that developers need in a cross-distro fashion, similarly to how Flatpak and Flathub work for graphical apps. Of course, long-term our goal is an OS for the broader public. We're working on this across a number of different fronts, including: Enablement for systemd-homed across the stack, allowing for stronger per-user disk encryption and other benefits. Experimental work in the kernel to solve a bunch of limitations we've run into with filesystems. Ada Magicat went on a deep dive on her recent All Systems Go talk! Deep integration between systemd-homed and the session's keyring, to increase security of the session's lock screen. Dhanuka Warusadura gave a lightning talk at All Systems Go about his work on this. A new feature-complete installer, to replace the current placeholder installer. Julian Sparber also gave a talk about this at All Systems Go, with a live demo! Support for delta updates in systemd-sysupdate, to avoid downloading a full 2 GB image every time. Better integration with the TPM, including a graphical TPM recovery UI. Currently, trying to type a TPM recovery key into Plymouth is rather difficult. Improved support for ARM hardware, including some specific models of phones. Abderrahim Kitouni gave a lightning talk at GUADEC this year about the current installation process. Shipping our pre-installed apps as Flatpaks, and reworking the app release process to unify the build definitions between GNOME OS, the GNOME Nightly Apps Flatpak repo, and Flathub. An STF project focused on Flatpak and Portals, which will enable broader app support on GNOME OS (like VPNs!). You can learn more from the announcement blog post. Some of these areas require lots more work before they're ready for prime time, and even once they're ready it's not clear that it'll be possible to migrate existing GNOME OS installations to use these new features. At FOSDEM this year, we discussed this issue and decided that we won't declare GNOME OS as "stable" until we can confidently commit to maintaining a clear long-term upgrade path for existing installations. However, we also don't think that all these things need to be in place before GNOME OS can be used by more people than it is today. It's not ready for the masses yet, but it is ready for early adopters that aren't quite adventerous enough to run a nightly OS. Introducing GNOME OS Preview GNOME OS Preview is the current GNOME OS Nightly, but without the "nightly" aspect. Instead, it's built from the latest stable GNOME release (GNOME 51.1, at the time of writing). GNOME OS Preview lives in the same branches on GitLab as the stable org.gnome.Platform Flatpak runtimes. GNOME OS Preview is only a few weeks old, so it has only seen limited real-life testing so far. We therefore expect people will uncover some issues initially. That said, Preview is significantly more stable than Nightly, which dozens of people have been daily driving without major issues for over a year. Thus, we're feeling confident that Preview is going to be reliable enough for daily use by non-developer enthusiasts. In all likelihood, GNOME OS Preview will be with us for a few years. However, we are explicitly not promising updates in

perpetuity. If you're not OK with re-installing your OS in a few years to switch to GNOME OS Stable, this is not for you yet. But if you've already been curious about giving GNOME OS a try, this is the moment! GNOME OS Preview is available for download on the newly-refreshed os.gnome.org! GNOME OS Nightly is still available from there too. As always, you can reach us on Matrix at #gnome-os:gnome.org or via the gnome-os tag on Discourse

- Matthew Garrett: Epic's EAC2 v2 seems fairly easily bypassable (2026/10/15 20:23)
  Epic just announced the upcoming version 2 of their Linux implementation of Easy Anti-Cheat. It's based on using the Linux eBPF security module to monitor what's happening, and then a bunch of extra infrastructure to make sure this can't be bypassed. The core part of this is requiring secure boot be enabled (ensuring that modified kernels can't be used), using measured boot to verify that secure boot is enabled and which keys are used (to prevent you from just signing your own modified kernel), and then doing remote attestation, a mechanism where the TPM generates a signed copy of the boot measurements and passes that off to a server which verifies it came from a real TPM, verifies that you have secure boot enabled and are using the right keys, and presumably then allows you into the game. This is all fine as long as you can prove that the measurements provided are real, and that's actually a lot harder than you might think. First, we can simply make up our own quote signed with our own key and say that it came from a TPM. This is avoided by every TPM having an Endorsement Key (or EK) and an associated certificate, with that certificate being signed by the TPM vendor. If you have a list of all the legitimate TPM vendors and their CA keys, you can verify that the measurements came from a real TPM. Easy enough. So we need to get the measurements from a real TPM, and the one on our motherboard contains the measurements that were generated by the firmware and bootloader, and we can't reset those at runtime without resetting the TPM, which… oh huh there's actually nothing stopping us from doing that at runtime, as long as we have access to the hardware. Many desktop boards have the TPM on a small plugin module with a standard pinout that includes a reset pin. Simply short that to ground briefly and the TPM is back in its initial state, and userland can write whatever measurements it wants to it. Boot a modified kernel which lies to the eBPF code? As long as you replay the expected log values from a boot with secure boot on and a trusted kernel, the TPM will still happily provide a legitimate set of measurements that'll satisfy the checker. What if the TPM is less accessible, either because it's mounted directly to the motherboard or it's inside the CPU or chipset? That's going to be trickier. But here's the thing. The quote needs to come from a TPM, but not necessarily the TPM on your motherboard. TPMs these days are usually connected via the Serial Peripheral Interface, usually referred to as SPI. Microchip make a USB to SPI bridge, and there's a Linux driver. Wire one of those up with an appropriate header and you can plug in an easily available TPM module intended for plugging into a motherboard, and now we tell the kernel to ignore the motherboard TPM and make /dev/tpmrm0 point at the USB one and program it with a set of trusted measurements and once again everything is going to look just fine (yes obviously the eBPF code can look for this, but we're running a modified kernel so we can just lie to the eBPF code). Remote attestation on PCs just doesn't work that well in environments where you can't associate a specific TPM with a specific system, which is basically everywhere except enterprises. But maybe that doesn't matter and cheaters wouldn't be willing to spend the $20 or so it'd take to produce one of these things? That would seem somewhat surprising, but it's not impossible. And obviously any server-side cheat detection that triggers could then block that specific TPM, meaning they'd need to go and get another TPM, and maybe that ends up being enough to discourage people from bothering. From a practical level, this may be sufficient. But at a technical level? It's not great.
- Jussi Pakkanen: Implementing a simple version of ThreadPoolExecutor in Pystd (2026/10/15 15:57)
  One of my favourite things in the Python standard library is the ThreadPoolExecutor. It is a very simple way to run a bunch of tasks in parallel,

such as converting or repacking a whole bunch of image files from one format to another. You write a pure function that takes two arguments, the input and output file names, and does the conversion. Then you submit a bunch of tasks (basically function and argument pairs) and the executor runs them with maximal parallelism.I decided to implement this in Pystd. To make things simpler it does not yet support futures for return values. Writing a thread pool executor needs some fundamental building blocks such as threads, mutexes and condition variables. Once those were implemented (and working), writing the actual "business logic" turned out to be fairly simple. In fact, it took less time than writing the PNG processing code with libpng.First you need to write this helper class to register you task for running. It also stores all arguments etc that the task needs to do its thing. Yes, this has a bit of that "icky 90s Java" feeling to it but OTOH it is fairly simple, has unambiguous ownership semantics and compiles massively faster than your typical template monstrosity.Using this is just a matter of globbing for png files in the input dir and submitting tasks for them.The UX is a bit verbose. It could probably be made nicer with a helper method or two. Still, not too shabby for something written by hand in a single afternoon.For those interested in details, the code for this program is in the pngtest branch.

- Christian Schaller: Using Artificial Intelligence to reverse engineer hardware (2026/10/15 13:28)
  After my mother passed away I ended up with a big box full of old film negatives containing photos from childhood family trips and similar. So I decided to buy a negative photo scanner and found that the Plustek 8200i was supported under Linux so I bought that. Only to discover that the chipset in the scanner had changed to one that wasn’t supported under Linux anymore. So I had some success making Linux configuration tools for some of my hardware using Claude code, like my Dell Camera controller app so I thought, maybe Claude can help me get a working SANE driver. I basically got stuck and got nowhere for the longest time and even basically gave up on the project for a few Months. Then I saw a post on X from Tom Crider aka Glorious Eggroll talking about how he had managed to fix a mouse driver using AI tooling. He basically described how he set it up so that the AI agent had full control of a Windows instance to capture data needed to fix the driver. So I pasted his tweet into Claude and asked it to help me set up a windows virtual machine to do the same things Tom as doing for my scanner. I think the combination of having the ability to run scans under windows and capture the USB data while doing so worked wonders, probably combined with improvements to Claude code made a huge difference. Within a few hours I had my first ever working scan under Linux, sure with some quality issues, but 100% recognizable as the image in the negative. Some days later I had a near perfect driver, including support for Infrared scanning ready to go, only to discover that in the time between me giving up and then trying again someone had committed code to SANE to support the scanner already…..it happens sometimes. I also used Claude to write an application for doing negative scanning, which I named Negative Vibes, thought that was a cute name. Anyway, with the success of my scanner effort I thought, I wonder if I can use this to finally fix the last issues with MIPI camera support on my laptop. So I told claude to re-use the harness I had created and try to fix the MIPI camera on my laptop by updating the kernel driver and libcamera as needed. I started the job in the morning and by the end of the day I had a much improved situation with patches for both the kernel and libcamera ready. So I thought that this might be worthwhile sharing with the world, and while anyone could start from scratch I did ask Claude to make the harness generic and document it to make the pathway for anyone else who wants to try this on their hardware a little easier. Writing a driver for undocumented hardware has always been tedious. You capture bus traffic, stare at hex dumps, and form hypotheses about what each register or control transfer does. You then test them one at a time, usually for weeks. None of it is conceptually hard, but it takes a lot of patience and skills. That kind of work suits an AI model well. It can read thousands of lines of USB capture, notice that a sequence repeats with one byte changing, and propose what that byte means. It can write the test to confirm it, and it doesn’t get bored. The human still decides what to build and checks that it works on real hardware, but the slog shrinks a lot. hw-harness Be aware that I did not use

hw-harness in this form for my projects as this was basically bespoke code at that time, so this project is an attempt to make the harness more generic and shareable from the scripts I was using while doing the work. What hw-harness is meant to help your Claude or Codex session with: 1. Record a reference session. Run the vendor’s Windows driver in a KVM virtual machine, with the real device passed through to the guest. 2. Capture from outside the VM. Record the traffic on the Linux host with usbmon. It sits below the passthrough layer, so the guest can’t see it. 3. Repeat the same operation on Linux with your own driver, and capture that the same way. 4. Diff the two traces. Both come from the same instrument, so they compare directly. The differences are your list of driver bugs. For devices with no bus to tap, such as an internal MIPI camera, the harness uses Windows-side inspection instead. That means INF files, registry data, ACPI tables and device properties. The approach should be fine in the sense it doesn’t rely on copying any code instead it observes what the hardware does and only extract configuration data from the Windows drivers, so every constant in the new driver traces back to something recorded. So for those interested I have 3 short descriptions of the projects I did and what was needed. Project 1: a SANE backend for the Plustek OpticFilm 8200i Comparing the driver output of two drivers available. The first project is a SANE backend for the newer Plustek OpticFilm 8200i. It uses the Genesys GL128 controller (USB 07b3:1825) and had no Linux support. The older model with the same name uses a GL845 and is already handled by SANE. The backend supports: Color, gray, infrared and color+infrared modes Resolutions from 300 to 7200 dpi, at 8 or 16 bits per channel Dust and scratch removal from the infrared channel An exposure multiplier for dense negatives It’s packaged as a Fedora RPM, built automatically on every push. I mostly tested it with my own Negative Vibes application as there are not that many options under Linux for negative scanning. Upstream SANE’s genesys backend also gained support for this scanner in September 2026, so I compared the two. Both resolve the same fine detail at 3600 dpi. Mine is faster at full resolution (34.6 s against 54.4 s) and supports infrared dust removal. Upstream offers more resolution steps. The webpage has an interactive side-by-side comparison you can check out. These are single-unit, single-frame results, so treat them as one data point. Check out the interactive page I made showing the current state of the two drivers, hopefully the discoveries I made can be used by the SANE project to fix some smaller items in the upstream driver. Anyway, I put this project up more as a proof of concept due to that since I expect everyone including myself will use the upstream driver once it reaches parity with this one. Project 2: Negative Vibes Full scan of negative completed including infrared cleaning of image. A driver alone isn’t much use if scanning negatives is still painful, so I built Negative Vibes. It’s a GNOME app, written in Rust, for scanning colour and black-and-white negatives and slides on any film-capable SANE scanner. The scan is kept exactly as the scanner delivered it: 16-bit, linear, with the infrared plane. Everything else is a stack of non-destructive settings on top, so you can change the look and export again. It includes: Density-space negative inversion with film base and dye-crosstalk correction Roll-wide settings with per-frame overrides Infrared and software dust removal IT8 calibration through ArgyllCMS Export to TIFF, JPEG, PNG and DNG, with ICC profiles and metadata It ships as a Flatpak that includes the scanner backend. It’s tested on the 8200i so far. The goal is to support any film scanner SANE supports. For that, SANE needs a standard way to expose infrared channels, which it doesn’t have yet. Project 3: Fixing the MIPI camera on my ThinkPad Showing MIPI output before and after fixes. Colors are better than they look as I was sitting in a relatively dark room by the end of the session. The camera in my Lenovo ThinkPad X1 Carbon Gen 14 (Aura Edition) uses a Sony IMX471 sensor behind an Intel IPU7. On Linux it produced dark images with heavy vignetting and a strong yellow-green cast. mipi-fixes-x1-aura collects the fixes. It’s only relevant to this laptop, which is why it isn’t one of the bigger packaged projects, but I have already submitted the kernel patches to Kate Hsuan and the libcamera patches is being reviewed by Wim Taymans for submission to Libcamera. Kernel (IMX471 driver): A crop-selection bug returned an error before storing the crop rectangle, so libcamera assumed the full pixel array. The test-pattern register was declared with the wrong width. The analogue gain cap was too low, which

cost about 1.8 stops of sensitivity. libcamera software ISP: Lens shading correction was missing entirely, leaving the corners about 2.7 stops darker. White-balance statistics were summed at the sensor’s native bit depth against an 8-bit black level. That skewed white balance, and the fix applies to any RAW10/RAW12 sensor. In a controlled low-light test, frame brightness rose from 31.1 to 90.9. The red/green and blue/green ratios moved from 0.890 and 0.672 to 1.047 and 0.823, close to a USB webcam reference. Blue is still about 17% short in low light, which I put down to grey-world white balance. Fedora 44 RPMs are on the releases page. What I learned Reference captures are a big step. Having the AI compare a known-good trace against a failing one turns a vague “it doesn’t work” into a concrete list of differences. For example for the MIPI camera having a USB camera also connected so Claude could compare the two captures was critical. The AI is good at the tedium. That covers decoding captures, writing probes and cross-checking constants. I still have to be the one holding the physical hardware. Real hardware is the only judge. Every claim above was checked on the actual scanner or laptop. If you have hardware with no Linux driver, try hw-harness. Thanks again to Glorious Eggroll for the inspiration. NOTE: I am not claiming that my scans or MIPI camera is perfect now, but the scanner gives me decent looking scans and the MIPI camera looks a lot better than before. So over time I am sure we be able to tweak and polish further, but I am happy that it allowed me to move the needle despite lacking the in-depth knowledge for this type of work. —

- Gedit Technology: gedit 51.0 released (2026/10/15 10:00)
  The gedit text editor version 51 has been released! A previous blog post, News, April-May 2026 already covers most of the changes. A reminder that gedit is still alive and is available to install, as an alternative. Progress has been made since May, but it's mostly under-the-hood changes, preparatory work to replace some internals and eventually make bigger visible changes to the application(s). B2B Services gedit is not only an application, there is a set of libgedit-* shared libraries underneath. This collection of re-usable code can be leveraged to create new products, and B2B services are possible. This was another reminder, but it cannot hurt ;-)
- Jakub Steiner: Digger (2026/10/15 00:00)
  The fact this movie tanked at the box office just makes the ride all the more enjoyable. I had a bad headache and had to call off the screening with a friend yesterday. Thursday is when the movies rotate in our part of the world where we use the un-American units, so I thought I'd missed yet another great cinematic experience. Luckily they still showed it in a small neighboring town, to the two of us and two other people, with an introduction by a charming young person. I don't remember anything these days, but I clearly remember when I saw Dr. Strangelove in Amherst, NH, in 1993. I was completely taken by the humor, despite thinking I'd never enjoy an ancient movie like that. It's probably way longer between my viewing and today than it was between then and the actual production of it. I don't wanna check. Anyhow, this is masterfully done cinema, and all four of us laughed out loud at the horrors we saw. Absolute must-see. ★★★★★
- Jakub Steiner: Pixels Wallpaper Update (2026/10/15 00:00)
  Usually I blog about new additions to the GNOME wallpapers after a release, to give some behind the scenes look on how they were produced. This time I'm asking for some feedback beforehand. Pixels has shipped as a wallpaper before. The wallpaper landed back in 2021 and keeps getting refreshed as new Circle apps join the lineup, and 47 added LCD and LCD-rainbow, revealing the curator's pixelart kink. The raw material is a side project years in the making — app icons redrawn as chunky pixels (Pixels, Mo' Pixels, Revert That Vector Nonsense!). The idea and layout remains the same as the old wallpaper. This time that's more than two variants though. The old Adwaita default used to cycle morning, day and night variants through a slideshow. While we switched to the dark/light variants, the format is still understood today. Thus pixelart comes in not 2, not 3, but 9 color variants. And here's where I'd like your feedback. Do you like the focus on contrast and having the whole set

with dark background colors? Should there be more bright colors in the mix? Do you think the grain goes against the spirit of the pixel art? Is the small grid too toned down with the depth fog? As always, the Blender project files are available in wallpaper-assets if you want to poke at the node setup yourself: As with any wallpaper, don't judge it based on a preview, regardless of how large you view it. Apply it, use it. The most straightforward way to test GNOME development is with GNOME OS. It ships pixels and all the other wallpapers.

- Lennart Poettering: Mastodon Stories for systemd v262 (2026/10/14 22:00)
  On September 22 we released systemd v262 into the wild. In the weeks leading up to that release (and since then) I have posted a series of serieses of posts to Mastodon about key new features in this release, under the #systemd262 hash tag. In case you aren't using Mastodon, but would like to read up, here's a list of all 35 posts: Post #1: Word List Based Hostname Patterns Post #2: Extended Attributes on Socket Inodes Post #3: Varlink Socket Tagging & varlinkctl list-sockets Post #4: Machine Tags from udev/hwdb Post #5: systemd-sysupdate's installdb Post #6: Signed systemd-report Reports Post #7: CLI Introspection via --introspect-cli Post #8: fscrypt v2 Support in systemd-homed Post #9: Read-Only ESP Support in systemd-boot/systemd-stub Post #10: Recovery Key Enrollment on First Boot & Varlink IPC for systemd-cryptenroll Post #11: Rogue TPM Protection for LUKS Enrollments Post #12: Encrypted + Verity Protected DDIs Post #13: FONT_SCALE= in /etc/vconsole.conf Post #14: Static, Single-Binary Service Manager Builds Post #15: Intel TDX CoCo Support in systemd-vmspawn Post #16: Varlink IPC for systemd-sysinstall Post #17: systemd-report as Varlink Service & New Metrics Post #18: Sessions as Inhibitors via pam_systemd's inhibit= Post #19: NvPCR Rework Post #20: bootctl link-auto Post #21: systemd-sysupdate Completion Hooks Post #22: /etc/clonetab Post #23: Atomically Enqueuing Multiple Jobs Post #24: Following the Journal via Varlink Post #25: Coredumps via the Kernel's New Socket Protocol Post #26: Hardware-Suggested systemd-sysupdate Features/Components Post #27: ActivatingConcurrencyMax= Post #28: Restarting/Reloading Units on confext/sysext Application Post #29: System Credentials for CoCo VMs in systemd-vmspawn Post #30: Persisting the TPM EK Post #31: systemd.firstboot=headless Post #32: RestartRandomizedDelaySec= Post #33: systemd.credentials_boot_policy= Post #34: systemctl kexec --kernel-cmdline-reuse Post #35: PCR Signature Policyrefs I intend to do a similar series of serieses of posts for the next systemd release (v263), hence if you haven't left tech Twitter for Mastodon yet, now is the opportunity. My series for v263 will begin in a few weeks most likely, under the #systemd263 hash tag. In case you are interested, here is the corresponding blog story for systemd v261, here for v260, here for v259, here for v258, here for v257, and here for v256.
- Juan Pablo Ugarte: Cambalache 1.2 Released (2026/10/14 20:25)
  Hello, I am pleased to announce a new Cambalache release. Cambalache is a WYSIWYG (What You See Is What You Get) tool that allows you to create and edit user interfaces for Gtk 4 and 3 applications. This new version depends on GNOME SDK 51 and includes all new properties and types from Gtk and Adwaita! It also removes the file hash from the project file which should prevent unnecessary git conflicts. In case you missed it, you can read more about Cambalache in version 1.0 release blog. Cambalache’s First Major Milestone! Help us continue developing If you find this application useful you can support it’s development by contributing on Liberapay or Patreon Where to get it? From Flathub flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo flatpak install flathub ar.xjuan.Cambalache or directly from gitlab git clone https://gitlab.gnome.org/jpu/cambalache.git Matrix channel Have any question? come chat with us at #cambalache:gnome.org Mastodon Follow me in Mastodon @xjuan to get news related to Cambalache development. As usual happy coding!
- Matthew Garrett: Linux Lockdown and Hibernation (again) (2026/10/13 18:14)
  (This is an updated version of an older post - the introduction remains the same, the technical details are different) Linux draws a distinction

between code running in kernel (kernel space) and applications running in userland (user space). This is enforced at the hardware level - in x86-speak1, kernel space code runs in ring 0 and user space code runs in ring 32. If you're running in ring 3 and you attempt to touch memory that's only accessible in ring 0, the hardware will raise a fault. No matter how privileged your ring 3 code, you don't get to touch ring 0. Kind of. In theory. Traditionally this wasn't well enforced. At the most basic level, since root can load kernel modules, you could just build a kernel module that performed any kernel modifications you wanted and then have root load it. Technically user space code wasn't modifying kernel space code, but the difference was pretty semantic rather than useful. But it got worse - root could also map memory ranges belonging to PCI devices3, and if the device could perform DMA you could just ask the device to overwrite bits of the kernel4. Or root could modify special CPU registers ("Model Specific Registers", or MSRs) that alter CPU behaviour via the /dev/msr interface, and compromise the kernel boundary that way. It turns out that there were a number of ways root was effectively equivalent to ring 0, and the boundary was more about reliability (ie, a process running as root that ends up misbehaving should still only be able to crash itself rather than taking down the kernel with it) than security. After all, if you were root you could just replace the on-disk kernel with a backdoored one and reboot. Going deeper, you could replace the bootloader with one that automatically injected backdoors into a legitimate kernel image. We didn't have any way to prevent this sort of thing, so attempting to harden the root/kernel boundary wasn't especially interesting. In 2012 Microsoft started requiring vendors ship systems with UEFI Secure Boot, a firmware feature that allowed5 systems to refuse to boot anything without an appropriate signature. This not only enabled the creation of a system that drew a strong boundary between root and kernel, it arguably required one - what's the point of restricting what the firmware will stick in ring 0 if root can just throw more code in there afterwards? What ended up as the Lockdown Linux Security Module provides the tooling for this, blocking userspace interfaces that can be used to modify the kernel and enforcing that any modules have a trusted signature. But that comes at something of a cost. Most of the features that Lockdown blocks are fairly niche, so the direct impact of having it enabled is small. Except that it also blocks hibernation6, and it turns out some people were using that. The obvious question is "what does hibernation have to do with keeping root out of kernel space", and the answer is a little convoluted and is tied into how Linux implements hibernation. Basically, Linux saves system state into the swap partition and modifies the header to indicate that there's a hibernation image there instead of swap. On the next boot, the kernel sees the header indicating that it's a hibernation image, copies the contents of the swap partition back into RAM, and then jumps back into the old kernel code. What ensures that the hibernation image was actually written out by the kernel? Absolutely nothing, which means a motivated attacker with root access could turn off swap, write a hibernation image to the swap partition themselves, and then reboot. The kernel would happily resume into the attacker's image, giving the attacker control over what gets copied back into kernel space. This is annoying, because normally when we think about attacks on swap we mitigate it by requiring an encrypted swap partition. But in this case, our attacker is root, and so already has access to the plaintext version of the swap partition. Disk encryption doesn't save us here. We need some way to verify that the hibernation image was written out by the kernel, not by root. And thankfully we have some tools for that. Trusted Platform Modules (TPMs) are cryptographic coprocessors7 capable of doing things like generating encryption keys and then encrypting things with them. You can ask a TPM to encrypt something with a key that's tied to that specific TPM - the OS has no access to the decryption key, and nor does any other TPM. So we can have the kernel generate an encryption key, encrypt part of the hibernation image with it, and then have the TPM encrypt it. We store the encrypted copy of the key in the hibernation image as well. On resume, the kernel reads the encrypted copy of the key, passes it to the TPM, gets the decrypted copy back and is able to verify the hibernation image. That's great! Except root can do exactly the same thing. This tells us the hibernation image was generated on this machine, but doesn't tell us that it was done by the kernel. We need some way

to be able to differentiate between keys that were generated in kernel and ones that were generated in userland. TPMs have the concept of “localities” (effectively privilege levels) that would be perfect for this. Userland is only able to access locality 0, so the kernel could simply use locality 1 to encrypt the key. Unfortunately, localities are typically only available on systems that implement a Dynamic Root of Trust Measurement like Intel’s TXT, and most consumer systems have no support for this. So, we need another approach. TPMs have a set of Platform Configuration Registers (PCRs), intended for keeping a record of system state. The OS isn’t able to modify the PCRs directly. Instead, the OS provides a cryptographic hash of some material to the TPM. The TPM takes the existing PCR value, appends the new hash to that, and then stores the hash of the combination in the PCR - a process called “extension”. This means that the new value of the TPM depends not only on the value of the new data, it depends on the previous value of the PCR - and, in turn, that previous value depended on its previous value, and so on. The only way to get to a specific PCR value is to either (a) break the hash algorithm, or (b) perform exactly the same sequence of writes. On system reset the PCRs go back to a known value, and the entire process starts again. Some PCRs are different. PCR 23, for example, can be reset back to its original value without resetting the system. We can make use of that. The first thing we need to do is to prevent userland from being able to reset or extend PCR 23 itself. All TPM accesses go through the kernel, so this is a simple matter of parsing the write before it’s sent to the TPM and returning an error if it’s a sensitive command that would touch PCR 23. We now know that any change in PCR 23’s state will be restricted to the kernel. When we encrypt material with the TPM, we can ask it to record the PCR state. This is given back to us as metadata accompanying the encrypted secret. Along with the metadata is an additional signature created by the TPM, which can be used to prove that the metadata is both legitimate and associated with this specific encrypted data. In our case, that means we know what the value of PCR 23 was when we encrypted the key. That means that if we simply extend PCR 23 with a known value in-kernel before encrypting our key, we can look at the value of PCR 23 in the metadata. If it matches, the key was encrypted by the kernel - userland can create its own key, but it has no way to extend PCR 23 to the appropriate value first. We now know that the key was generated by the kernel. Unfortunately it turns out that some people actually make use of PCR 23, and restricting it to the kernel makes them sad. There are other resettable PCRs, but only in non-0 localities, so (a) we can’t reset them on most systems and (b) we’ll just find someone who uses them anyway and run into the same problem. We need a different approach. Thankfully TPMs have a small chunk of non-volatile RAM that can be used for storing things, and by “storing” I mean they can be used in a range of different ways. One of these is effectively as additional PCRs. These values can be configured such that they work the same way as PCRs, with the only way to modify them being to append a new hash to their existing value and update the new value to the hash of that merged value. We can allocate one of these for in-kernel use and reject any access from userspace, and also there are officially allocated ranges (so people shouldn’t be using random values) and the namespace is large enough anyway that w eshouldn’t collide with anyone, so we can carve one of those out and use it for this purpose. Unfortunately, though, these NV indexes aren’t included in the creation data we get when we create a key, so we need another approach. TPMs have the concept of sessions, a series of commands executed in sequence. A flag allows these sessions to be “audited”, with the TPM keeping a running track of the hash of the commands executed and the results returned. At the end of the session you can ask for a signed copy of this hash, and if you’ve kept a log of all the commands then you can verify that they match that signed hash and know that this log represents a real communication with the TPM. If we read the NV index and generate the key inside the same session, we can verify that the key we’re looking at was created while the NV index had a specific value, and if only the kernel can set that value then we know that the key was generated by the kernel. Or, well, can we? Only new kernels would impose the restriction on modifying that NV index. An attacker could boot an old kernel, generate a key there in userland, sign a hibernation image with it, and then reboot into a new kernel.

The new kernel would see a valid signature with a valid NV index associated with it and have no way to know it was generated from userland rather than the kernel. We need to be able to tell the difference between kernels that implement this feature and ones that don't, and we need to do that without being able to go back in time to change old kernel behaviour. We can extend a PCR to indicate we implement the new functionality, but so can an attacker running an old kernel. The normal approach to determining whether an event occurred in a specific phase of the boot process is to "cap" the PCRs - extend them with a known value that indicates a transition between stages of the boot process. Any events that occur before the cap event must have occurred in the previous stage of boot, and since the final PCR value depends on the order of measurements and not just the contents of those measurements, if a PCR is capped before userland runs, userland can't fake the same PCR value afterwards. If Linux capped the PCRs before userland started running, we'd be able to place a measurement there before the cap occurred and then prove that that extension occurred before userland had the opportunity to interfere. We could simply place a statement that the kernel supported the NV index restrictions there, and we'd be fine. Unfortunately Linux doesn't currently do this, and adding support for doing so doesn't fix the problem - if an attacker boots a kernel that doesn't cap a PCR, they can just cap it themselves from userland. So, we're faced with the same problem: booting an older kernel allows the system to be placed in an identical state to the current kernel, and a fake hibernation image can be written out. Solving this requires a PCR that was being modified after kernel code was running, but before userland was started, even with existing kernels. Thankfully, there is one! PCR 5 is defined as containing measurements related to boot management configuration and data. One of the measurements it contains is the result of the UEFI ExitBootServices() call. ExitBootServices() is called at the transition from the UEFI boot environment to the running OS, and the kernel contains code that executes before it. So, if we measure an assertion regarding whether or not we support restricted access to NV indexes into PCR 5 before we call ExitBootServices(), this will prevent userspace from spoofing us (because userspace will only be able to extend PCR 5 after the firmware extended PCR 5 in response to ExitBootServices() being called). Obviously this depends on the firmware actually performing the PCR 5 extension when ExitBootServices() is called, but if firmware's out of spec then I don't think there's any real expectation of it being secure enough for any of this to buy you anything anyway. That's what this patchset does. Review feedback so far seems positive, although James Bottomley suggested a design improvement. My approach of using an NV index as a PCR and generating a signing key made sense as a consequence of my original approach involving PCR 23, which had been to use the key creation data and not use audit sessions at all. But the goal isn't to use a key - it's to prove that hibernation occurred in-kernel, and while signing is one way of doing that, there are others. His suggestion is to simply record the hash of the hibernation image into the NV index and then read that back as part of the audit session. On resume we verify the audit session log is legitimate, read the hibernation image, and compare the hash of the hibernation image against the hash in the log. If it matches, and if the PCR 5 value matches, we know the audit session happened in kernel and that the hibernation image hasn't been modified. It still seems like we're missing something, though. If we're not storing a key anywhere, how do we verify the audit session? To explain this we're going to need to go into some details about how TPM keys work in the first place. TPMs can produce a special kind of key called an Attestation Key, or AK. Attestation keys are used to sign attestation statements from the TPM, such as PCR quotes or audit logs. We need a separate key type for this in order to prove that the signature was generated as part of an attestation - normal TPM signing keys are fundamentally designed to sign arbitrary material, and so can be used by userland to sign something that looks identical to an attestation record. By being able to distinguish the types of key the TPM uses, we can prove that a signature was generated using an AK, and so know that it's a real TPM attestation (because nothing other than the TPM can sign stuff with an AK). But even beyond this distinction, TPMs have different key types. Ordinary keys are generated using an RNG and will be unique. Primary keys are generated using a

seed that is used as an input into a key derivation function. For a given TPM, using the same seed to generate the same type of primary key will always generate the same primary key8. We can have the kernel deterministically generate an AK during suspend in order to sign the audit log, and then generate the same key again on resume to validate it. This means we don't need any other form of trusted storage for the AK - the TPM already provides that functionality. This is all rather complicated but I've been thinking about this for some time now and I haven't come up with a cleaner solution. The only real open issue is the degree to which machines actually perform the PCR 5 extension. If it turns out there's a bunch that don't we'll need to come up with some alternative approach to identifying whether the kernel supports this functionality, but the design is such that that should be invisible to userspace so adding alternatives later shouldn't be an issue. Filtering accesses to the NV index is also not super clean, and depends on there not being alternative mechanisms to get at the TPM by sending raw commands. Again, this is intended to be opaque to userland and can be improved if we come up with something better later on (such as a secure way to pass a secret to the kernel on boot that can be used as an auth value for acess to the NV index). Anyway. Time to go and fix the next thing. Other architectures use different terminology here, such as "supervisor" and "user" mode, but it's broadly equivalent ↩ In theory rings 1 and 2 would allow you to run drivers with privileges somewhere between full kernel access and userland applications, but in reality we just don't talk about them in polite company ↩ This is how graphics worked in Linux before kernel modesetting turned up. XFree86 would just map your GPU's registers into userland and poke them directly. This was not a huge win for stability ↩ IOMMUs can help you here, by restricting the memory PCI devices can DMA to or from. The kernel then gets to allocate ranges for device buffers and configure the IOMMU such that the device can't DMA to anything else. Except that region of memory may still contain sensitive material such as function pointers, and attacks like this can still cause you problems as a result. ↩ This describes why I'm using "allowed" rather than "required" here ↩ Saving the system state to disk and powering down the platform entirely - significantly slower than suspending the system while keeping state in RAM, but also resilient against the system losing power. ↩ With some handwaving around "coprocessor". TPMs can't be part of the OS or the system firmware, but they don't technically need to be an independent component. Intel have a TPM implementation that runs on the Management Engine, a separate processor built into the motherboard chipset. AMD have one that runs on the Platform Security Processor, a small ARM core built into their CPU. Various ARM implementations run a TPM in Trustzone, a special CPU mode that (in theory) is able to access resources that are entirely blocked off from anything running in the OS, kernel or otherwise. ↩ Unless the TPM is cleared, in which case it'll generate a new internal seed and now the keys will be different to before the clear ↩

- Jiri Eischmann: Fedora & CentOS at LinuxDays 2026 (2026/10/13 11:06)
  LinuxDays 2026, the largest open-source event in Czechia, took place on the first weekend of October, and we had a Fedora & CentOS booth there again. This time, we got a smaller table and our booth equipment was more modest because no one traveled from Brno by car—we only had what we could carry on the train. However, we had quite a few nice stickers and one Framework laptop, which generated a lot of attention again. I was happy that I didn't have to staff the booth alone; I was accompanied by Vojtěch Trefny, a long-time Fedora contributor. Even though CentOS Stream has existed for seven years now, we still received questions like: "Oh, CentOS is still alive? I thought Red Hat killed it a long time ago." This was typically the starting point for a conversation about the differences between the original CentOS and CentOS Stream, as well as their relation to Red Hat Enterprise Linux. Most questions about Fedora were related to Framework, which didn't come as a surprise given that we were showcasing Fedora Linux 45 on a Framework Laptop 16. People asked whether there is any cooperation between Fedora and Framework, whether Fedora Linux is a good OS for Framework hardware, and whether Framework is a good laptop option for Fedora Linux. To all of that, we had very positive answers. It was an exhausting weekend because we had dozens of conversations, but it was also refreshing to meet Fedora and

CentOS users and hear feedback directly from them. I'd like to thank Fedora for sponsoring my trip and Vojtěch for helping me with the booth.

- Justin Wheeler: Managing multiple Git repos with AI: How I keep up with Fedora's sprawl (2026/10/13 08:00)
  No single person can know everything happening in Fedora at any given time. I say this often in our community, half as a warning and half for comfort. For a while, I did not really believe it applied to me. I wanted to know everything. I joined new teams, participated in tickets, and either participated or led meetings across the project. Surprisingly, as a young university student, I held a vast amount of it in my head. But Fedora does not stand still long enough for that to last. Eventually, I was spread thin across more teams than I could realistically track. As my academic load increased or I began an internship, promises I made quietly fell off my radar. It was not because I did not care. It was because the systems I used to track them could not keep up with the sprawl, when my available time to contribute changed. However, as I reentered the Fedora community as a Red Hat employee, the same systems I used as a student were challenged. As I get going with my new role, I am rethinking how I can rebalance my bandwidth sustainably. So, this post is about how I started managing multiple Git repos with AI agents to claw back some of that bandwidth. It is not really a story about a remarkable tool. It is an argument that in a project this sprawling, protecting your attention is the real win. I will cover two moments when things fell through the cracks, the workflow I built to catch them, a real example of it working, and an honest take on trusting AI when it sometimes gets things wrong. Why managing multiple Git repos gets hard at Fedora's scale Fedora is sprawling. Fedora's distribution Git (dist-git), the home for package sources, holds something like 20,000 packages. Add the team and SIG repositories that coordinate the people behind those packages on Fedora Forge, and the number of places where decisions get made is enormous. Yet, most contributors only touch a handful of them. However, even a handful is often scattered across different platforms, with no single view of all of it. Where things fell through the cracks Before this workflow, it felt like pure chaos, and I have the scars to prove it. Fedora runs on mailing lists, Fedora Discussion topics, Matrix chat rooms, and Fedora Forge issues/tickets. Furthermore, all of these are spread across dozens of teams, Working Groups (WGs), and Special Interest Groups (SIGs). In short, there are a lot of different places to follow a lot of different things. For a long stretch, the Fedora Marketing Team had a backlog of tickets. In a few cases, contributors asked for talks and presentations to be uploaded to the Fedora YouTube channel. I had the access to do it and other YouTube maintainers were no longer active, so therefore, it was quietly mine to own. I saw the tickets at least once, but I was overworked and busy with higher-priority work. The YouTube videos slid off my list. A new volunteer eventually picked them up months or even a year later. Additionally, Fedora also runs meetings well… on paper. Fedora holds virtual text meetings in Matrix chat rooms with a bot that logs everything. Each meeting has a clear start, a clear end, and annotated topics. But a faithful log is not the same as follow-through. For example, in AI/ML SIG meetings, we discuss a decision or someone takes an action item, and it gets logged. But it does not always make it back into the relevant ticket or discussion post. The record exists, but it does not connect itself to the places that need updating. Patterns like this repeat themselves. The examples above are merely specific examples of where things fall through the cracks. However, there are things like this that happen everyday in different corners of the community too. The AI multi-repo workflow So, here is the workflow I use now. It does not get rid of all the chaos, but it helps me manage it much better. First, I point an LLM-gen-AI agent at a directory holding several cloned Git repositories. For example, the AI/ML SIG docs repository, AI/ML SIG tickets repository, and Agent Skills library. Then I ask it to analyze across all of them. I also feed it meeting logs or summaries. I ask it to cross-check them against open tickets and Fedora Discussion topics: Did we document the decisions? Did we track the action items? Is anyone still waiting on a reply? Critically, I do not want it reading only files from my local git clones. I want it to query the remotes too: the Fedora Forge API, issues, discussions, and more. The underlying Forgejo API is enormous. It lets the agent track the actual cadence a team works at, not just a snapshot. A word on trust and

hallucination I have to be honest here: the agent sometimes gets things wrong. It will occasionally misread a thread or hallucinate a detail. However, I do not think that disqualifies it. Humans get things wrong too. I would rather have slightly imperfect visibility into tens, hundreds, or even thousands of repositories than none at all. Ultimately, the question is not whether the agent is ever wrong. The question is what level of risk you are willing to accept for the visibility you gain. A real example: Reviewer guidelines across three repos Our Fedora Agent Skills Library needed reviewer guidelines. Laura Barcziová (@lbarczio), on our reviewer team, wrote a solid REVIEWING.md in the ai-ml/skills-library repo. But our separate ai-ml/docs repository only had a page on creating a skill, and nothing on reviewing one. I used an agent in OpenCode to pull discussion context from Fedora Forge issues about our reviewer process. It helped me reconcile that context with Laura's draft. Then I orchestrated multiple Pull Requests across the repositories at once. This brought the AI/ML SIG docs site in line with what the team had actually decided. The real payoff is not the tool In conclusion, the payoff is that it hands me back the capacity to zoom out. In a project where nobody can hold it all in their head, something should watch dozens of repositories. It surfaces what changed and what needs to catch up. That means I can engage with people's feedback instead of just surviving the volume of it. When someone takes the time to review my Pull Request, I want the bandwidth to genuinely consider what they said. I do not want to just clear my queue. I hope this overview of my flow and process is insightful. As of yet, I do not have a commenting feature on my blog, although I am looking into this. Tag me in a LinkedIn post and we can continue the conversation there. Original photo by DM David on Unsplash. Modified by Justin Wheeler. Modified cover photo exclusively distributed under CC BY-SA 4.0.

- Javad Rahmatzadeh: Call for Volunteers to Help GNOME Extensions Team (2026/10/12 23:37)
  I remember back in 2020, people were complaining that we didn't have enough documentation for creating GNOME Shell extensions. They said it was almost impossible to figure out where to start. They were right, learning all of it without proper documentation was a huge challenge for me too. So, I decided to learn the GNOME Shell source code and shared what I learned in a YouTube tutorial series. I spent countless hours learning and making those videos, and the community was very happy to finally have guides and documentation. On the GNOME side, Sriram Ramkrishna started the GNOME Extensions Rebooted project. When he noticed my tutorials, he invited me to join the team to help write documentation. I spent many hours learning GTK4 and the GNOME Shell 40 source code to write the famous GNOME Shell 40 Port Guide. I'd argue that many extension authors would have given up without it. Migrating from GTK3 to GTK4 and adapting to the changes between 3.38 and 40 was overwhelming. I wrote the guide with many examples, so developers could easily understand how to port their code. Today we have all the documentation needed to start and maintain extensions, but nowadays, people are just simply vibe coding GNOME Shell extensions without reading any of it before submitting it to EGO. To be honest, in the age of LLMs, it feels like our documentation is feeding LLMs more than actual human developers. For years, I tried my best to help the community, answering questions daily and reviewing extensions submitted to EGO, often spending 4 to 10 hours a day managing the whole extensions eco system. GNOME was incredibly generous this year and awarded me the GNOME Pants Award 2026 for my contributions to GNOME and the extensions ecosystem. That was a major personal milestone, and honestly, a testament to the growth of our community as a whole. I'm writing this post because since early 2025, the volume of submissions has exploded. The vast majority are AI generated, and the huge workload is pushing me close to burnout. To be clear, all of my work is unpaid. While managing this volume easily matches a full-time job, my monthly donations remain close to zero. Given how many people use GNOME extensions daily, I truly believe our community can help make this workload doable. If you find value in what I do or if my work has helped you out over the years, please consider supporting me through donations. Without financial support, I simply won't be able to keep going at this pace. AI generated

extensions, especially when prompted by non developers are frequently bloated and poorly designed. Because extensions run directly inside the GNOME Shell process, quality control is critical. Low-quality, AI generated code directly threatens the stability of the entire shell, and vibe coders need to realize that. As someone who reviews code across all of EGO, I can tell you that LLMs produce needlessly complex, hard to read code that takes significantly longer to review. To make matters worse, some submitters rely on AI to generate replies during the review process. I even received a response that accidentally pasted the LLM's instructions intended for the user: Then, respond to the specific points it points to and maintain a real public repository, with testing instructions and development history. This gives the base reviewer to evaluate maintenance and quality, without trying to hide anything. On a positive note, a while back I published An Open Letter to AI for Generating GNOME Shell Extensions to fix specific code patterns their models were generating incorrectly and I noticed most of those issues disappear just a few weeks after that post, which is great news, but also proof of just how heavily our documentation is feeding these LLMs. Anyway, we need viable solutions to keep the EGO ecosystem stable, trustworthy, and sustainable: I'm asking developers who understand GTK and GNOME Shell source code to join the review team. I am happy to mentor and teach you the review process. Please note that while this is a volunteer role, we need people who can be consistently available during the week. As many of you know, we introduced the Shexli linter a few months ago to make the review process easier. We need someone to help us improve it. The workflow is straightforward: reviewers identify false negatives and false positives during reviews, and you fix them in Shexli. If you are interested in joining the review team or helping with Shexli, please reach out to us on GNOME Extensions Matrix channel. Thanks for being part of the GNOME community!

- Hylke Bons: Icon for Ripples (2026/10/12 00:00)
  Week 33 This week's icon is for Tau Gärtli's project: Ripples: "Rip CDs" Check out all weekly app icons created so far in the gallery and follow my icon creation adventures as they happen (including sketches) on the Fediverse. Need icons? I love designing icons and am happy to contribute them free of charge when your project is Free and Open Source. Join as a community sponsor to maintain a steady supply of app icons to the Linux ecosystem (every little helps!).
- This Week in GNOME: #269 Permission Admission (2026/10/09 20:09)
  Update on what happened across the GNOME project in the week from October 2 to October 9. GNOME Circle Apps and Libraries Brage Fuglseth reports This week Concessio was accepted into GNOME Circle. Concessio helps you understand and convert between unix permissions representations. Congratulations! Ronnie Nissan reports This week I released Concessio version 1.0.0. I am very proud that it got accepted into GNOME circle as announced above, it also got the following improvements. A full rewrite from Gjs to Vala, making Concessio very fast, the code base clean and easier to contribute to. Redesigned the file opening workflow Added drag and drop support for opening files Added support for changing file permissions, with the ability to undo changes Improved the umask calculator to calculate from umask, file, or directory permissions Improved accessibility throughout the app Improved input validation and editing behavior Updated translations I am also working on writing libacl binding for vala so I can add ACLs support to Concessio later on, if you have experience with writing bindings and this project interests you, please reach out. Thanks! Third Party Projects Ronnie Nissan says This week I also updated my other apps, Sitra and Embellish. They both got the latest GNOME runtime 51, Embellish also was also updated to NerdFonts v3.5.1 which adds the messing Devicons that were promised in v3.5.0. I would also like to announce that I am looking for a maintainer for Sitra, if anyone is interested in a Vala GTK4+Libadwaita program, please reach out, if I find you are willing to take care of the project, I will hand it over completely. Tanay announces Whisp V1.6.0 -Runtime Update and Fixes New Features Note Statistics (::count): Type ::count anywhere in your note to instantly inspect word, character, and line count statistics. Asterisk

Lists: Create and format bulleted lists using standard asterisks (*) alongside dash lists. Shortcut Recorder Improvements: Pressing Backspace inside the shortcut recorder now cleanly clears or disables assigned shortcuts. Fixes & Maintenance List Toggling Refactor: Fixed Ctrl+S list item toggling to prevent multi-checkbox stacking and unintended line deletions. GNOME 51 Runtime: Updated Flatpak runtime to GNOME 51. Slate Mode Polish: Fixed update notification banner layout in Slate Mode. Compose Key Fix: Corrected key press handling for Compose Key combinations. Ubuntu Icon & D-Bus Fixes: Fixed missing application icon issues on Ubuntu desktop sessions. Links : Download | Github | Website | Donate Whisp Development is made possible by users like you, If Whisp brings you value do consider donating. tfuxu announces Flood It 3.0 released, finally bringing resizable game boards and toggable number indicators on colors. Check it out on Flathub! Titouan Real reports Era 0.2.0 is out. Download Era from Flathub now and report any bugs you can find. Era is a calendar app for mobile and desktop, designed to stay simple for everyday use while offering powerful tools when you need them. It syncs with most online calendar providers and works offline, too. What’s new: Move events with drag and drop within the month view, or drag an event into another app to export or share it as an ICS file. Import external calendar events with high-performance parsing. If Era 0.1.0 felt slow, give 0.2.0 a spin! Major performance optimizations are live in this release, with even more speedups currently in development. Era is also available on GNOME Nightly for the latest features, bug fixes, and performance improvements. Anton Isaiev announces RustConn is a connection manager for SSH, RDP, VNC, SPICE, Telnet, Serial, Kubernetes, Web, and Zero Trust, built with GTK4 and Libadwaita. Version 0.23 is out, bringing major interface updates, workflow enhancements, and security hardening since 0.22: User Interface & Experience: Rebuilt the main window layout around the GNOME HIG with separate sidebar and content pane header bars (matching Files and Settings). Protocol filters are now neatly folded behind a single Filter button. True fullscreen mode now hides both the header bar and tab bar for an unobstructed session view. Added keyboard passthrough to hand desktop shortcuts (such as Super and Alt+Tab) directly to embedded RDP and VNC sessions. Connection & Session Management: Keystrokes can now be broadcast across groups of tabs, not just individual split panes. Any pane within a split can now be reconnected in place independently. External FreeRDP and VNC viewers can now be pinned on a per-connection basis. Zero Trust sessions with expired cloud logins now provide a one-click re-authentication flow (AWS, Google Cloud, Azure) directly within the tab. Added per-connection command macros triggered by keyboard shortcuts. Clusters can now automatically collect members matching a regular expression. Context menus now feature a dynamic Copy submenu showing only existing connection details, alongside an Edit Connection action. Credentials & Protocols: Added a read-only mode and search-from-root toggle for secret backends (Bitwarden, 1Password, Passbolt, pass, KeePass), allowing vault discovery without granting write access. Added support for KeePass databases protected by YubiKey Challenge-Response slots, complete with interactive “touch your key” prompts. Connection editor can now auto-detect a host’s MAC address for Wake-on-LAN in one click. Added opt-in H.264 support for RDP via Cisco OpenH264. Added a dedicated KDC address field for Kerberos authentication on realmd/sssd environments. Security Hardening: Multi-line terminal pastes now require confirmation via an interactive preview, preventing pastejacking attacks. Added route-change warnings: RustConn alerts before sending stored credentials if a connection’s host, port, account, or jump host changed since the last run. Hardened embedded RDP file downloads against hidden dotfiles (e.g., .bash_profile) and bidirectional-override character exploits in filenames. Centralized argument sanitization for FreeRDP launches to drop unauthorized parameters (e.g., /shell:, /proxy:) from imported profiles. Plus numerous stability improvements and bug fixes. Thanks to everyone who uses RustConn, reports bugs, contributes or supports the project - this cycle was driven largely by issues people filed. If you’d like to support development, the repo has a Sponsor link. https://github.com/totoshko88/RustConn https://flathub.org/apps/io.github.totoshko88.RustConn https://snapcraft.io/rustconn/ Rat Cornu announces Ratic got a new version 0.4.4 this week, with two big changes! First, the dynamic background

(based on the current music cover image) does not mess with colors anymore. It is much cleaner and consistent thanks to the color thief algorithm. Second, lyrics are now supported in ratic, both synced and unsynced! For synced lyrics, the current line is highlighted to follow your song, and you can click on the line you want to jump on to seek this particular position in the music. You can check the new version on flathub, on the repo, or even come discuss with us on our matrix room! Thanks again for all the people that contribute to the localization on weblate! I never imagined that so many languages would be added to a small project like mine. Shell Extensions swink announces Now Playing Card got several updates. It adds an animated indicator to the top bar with a compact media card for any MPRIS player: cover art, a seekable progress bar and transport controls. Recent updates (1.0.4–1.0.8) bring: a spectrum analyzer that moves with the sound itself a redesigned popup: one surface, rounded cover art, slimmer controls, and separate tiles when several players are running covers given as web addresses (e.g. Spotify) now load without an empty square while they download a right-click menu on the panel button to choose visibility and open settings click the speaker to mute, click again to restore the volume the card follows the shell theme's text colour, so light themes on a dark system (and the reverse) look right Source code is on GitHub. GNOME Fellowship Peter Eisenmann says I posted my monthly GNOME Fellowship update for the "slow" month of September https://blogs.gnome.org/p3732/fellowship-report-september-2026-moria/ Sophie (she/her) announces CMYK support in glycin and more – all in my September report for my GNOME Fellowship. Sovereign Tech Agency Philipp Sauberzweig reports I've blogged about another month of GNOME Design and Community Management as part of my Sovereign Tech Fellowship. I reflect on how the Fellowship has changed the conditions for my contributions, share updates on the GNOME Circle program, and show some exciting progress on standardising Scale Row and improving Calendar. Check out the full blog post! That's all for this week! See you next week, and be sure to stop by #thisweek:gnome.org with updates on your own projects!

- Peter Eisenmann: Fellowship Report September 2026 (Moria) (2026/10/09 12:03)
  I wanted to start this post with saying September was a bit of a slower month, as I was on vacation for a good chunk of it, but turns out that still quit some things happened Nautilus The pathbar completion improvements I mentioned in the last post landed. Completions are now reused when in the same directory. On top of that, further suggestions are now directly shown when accepting a previous suggestion. https://blogs.gnome.org/p3732/files/2026/10/pathbar-completion.webm I also created a (somewhat hacky) solution to immediately start completing when navigating to the end of the entry, which did not land yet. I also finished the (initial) graphical app chooser changes, which now has a "Show All" tile. This needs design input, but testing turns out to be a bit tricky as the Flatpak version uses the app chooser portal. Meanwhile, the code cleanup part of the app chooser rework already landed after Khalid Abu Shawarib reviewed it. In general Khalid reviews most of my nautilus MR, which I am very thankful for. Other than that I… Created an MR that adds support for the star column in Recent and Search and also allows hiding the column altogether. Fixed up a cleanup of the audio/video extension plugin, which drops custom content type lists and gives it a more generic name. Landed more code cleanups related to how file metadata is set. Made use of SPDX licence specifiers consistently across the whole project. Continued work on the xdg-user-dirs replacement MR. Sushi As sushi is (currently) only ever presented as a dialog to nautilus, it didn't need an app icon in the past, but with modern app distribution approaches (it's installable via Flathub) that no longer applies. I filed an app icon request some months ago, in which Jakub Steiner and I jammed up a shiny new icon togehter: My composition suggestion Jakub's refined version Sushi tries to show cover art when possible. This also included the capability to download cover art (something I fixed in July but seemingly forgot to mention). The downloaded covers were not stored in the file – a previewer shouldn't modify files after all – so it was a rather obscure feature to have. Fetching covers of course involved sending queries to other people's computers, which

might be unexpected default behavior, especially since html previews started asking for confirmation before fetching web content. Thus, Tau Gärtli and I agreed on removing this niche feature. As a replacement, I implemented something that seems more useful to me – support for cover art files that are in the same folder as the song, such as folder.jpg or cover.png. Sometimes file previewing fails due to various reasons (missing codecs, corrupted file, …). Instead of the previous dedicated page, I adjusted sushi's error handling so that basic file information is still shown, while the error is indicated by a banner: Other small changes: I fixed a bug of my own creation that prevented some mouse interactions. The fix is included in the 51.1 release that landed on Flathub. Sushi now uses the default .Devel developer version suffix Renderer loading times are limited now, preventing infinite load screens gnome-autoar Libarchive, the library used by gnome-autoar supports multi-threaded compression for some file types (7zip, xz, zstd), so I added an option to enable this in gnome-autoar. If all goes well nautilus will make use of this soon. I also fixed a signal parameter bug that was blocking a gnome-shell MR. The cause was that gnome-autoar correctly emits its signals in the main thread, but didn't wait for responses in the asynchronous thread that initiated the signaling. For simple fire-and-forget signals that wouldn't be an issue, but not only do some of gnome-autoar's signals have return values, there even is one particularly peculiar case of a signal with an out parameter – something I couldn't find another example of in other projects. I also created a handful of cleanup MRs, which bring the repository up to the standard of a modern (GObject-based) C project: a gobject-linter CI job, SPDX licence specifiers, G_LOG_DOMAIN usage, code cleanups and more. Roadmap Support the GNOME Project The GNOME Fellowships are funded by our community. If you would like to help the GNOME project to stay sustainable, please consider donating. AI Summary Freed from his shackles, Autoar learns that they all along had the capability to do multiple things at the same time. With this, cleaning up the library is no problem and even the strangely wound looking glass on the signaling lookout gets fixed. Through it the group can be seen moving into the depths of the gnome caves. With them they carry rations in the form of paper scrolls, depicting such accurate portrays of sushi that fiction and reality is not distinguishable to mortal digestion systems. Important note: All statements in this blog are fictional.

- Philipp Sauberzweig: Fellowship Update, 2026-10-07 (2026/10/07 13:21)
  It has been another productive month working on GNOME Design and Community Management as part of my Sovereign Tech Fellowship, and I want to share some of the most interesting developments. Introduction I want to begin by highlighting how much the Sovereign Tech Fellowship has changed the conditions under which I contribute to GNOME. As a volunteer, I had to fit my contributions around my day job, working on GNOME in the evenings and during holidays. The Fellowship has enabled me to significantly expand the scope of my work. I can focus more on platform-level design and work on structural changes that make it easier for others to contribute. This type of work reaches many people, and it makes me feel particularly effective. At the same time, working on GNOME is now my job, which allows me to maintain a much clearer separation between work and free time. I am really enjoying having more time for sports, and I have also started taking regular piano lessons again. I spent the last two weeks of August on holiday in nature and returned feeling refreshed and motivated. Thank you very much to the Sovereign Tech Agency for this opportunity! GNOME Circle GNOME Circle is a certification for high-quality third-party apps that also serves as an educational program for app developers and designers. Unfortunately, GNOME Circle had to close applications in May because the small team of volunteer reviewers could not keep up with the large backlog of unreviewed apps. Thankfully, several people responded to the call for reviewers, and Tobias recently introduced Sjoerd, Will, FineFindus and me to the review process. Sjoerd was able to start his first reviews of Concessio and Iconic immediately, drawing on the design experience he gained from receiving the Circle review of his app, Graphs. I also started reviewing two apps: Quadrapassel, a Tetris clone, and Mecalin, a keyboard typing lesson app. Quadrapassel is developed by Will, and my review aims to share some

of my design experience before he starts reviewing submissions himself. FineFindus has started reviewing Field Monitor, and Tobias and I provided feedback on their review before it was posted publicly. This represents a significant increase in our review capacity, and I am optimistic that we will soon complete the remaining reviews and then be able to reopen submissions. I would like to thank all previous and current Circle Committee members for setting up and contributing to this program, as well as for reviewing all the excellent GNOME Circle apps. If you want to get involved, reach out on the GNOME Circle Matrix channel. Scale Row Standardization Libadwaita provides several types of boxed-list rows for common use cases, such as text entries, password entries, switches, and dropdowns. However, it has so far been missing a row with a scale, which has led apps to implement custom widgets with inconsistent styling. Let’s change that. I started the Scale Row design process by collecting screenshots of existing use cases and examples of state-of-the-art designs from other platforms. Looking at other platforms is always helpful, but I also needed to take into account the requirements of our design system and its technical capabilities. Because Scale Row will be based on Action Row, it needs to support a title, subtitle, prefix widgets, and suffix widgets. To make the layout work in narrow windows, it also became clear that the scale should be placed below the title. Several rounds of sketching, mockups, and discussion led to the following preliminary design. Preliminary Scale Row design applied to common use cases Originally, I intended to design only the row layout and use the scale as it was. However, aligning the start and end of the scale with the labels and icons required us to change the scale’s padding. That change has already landed in Libadwaita, and we decided to take the opportunity to review the scale’s style as well. I found that the current style does not follow our design system’s 6-pixel grid, and that the knob is smaller than the minimum target size required by the WCAG accessibility guidelines. I therefore proposed several options inspired by other design systems. What we can adjust is largely restricted by what GtkScale technically supports, and I believe we may settle on the proposed pill-shaped style. The pill-shaped knob satisfies the WCAG requirements because a 24-pixel-diameter circle overlaid at the center of its bounding box does not intersect with adjacent targets, such as the scale’s trough. At the same time, the pill-shaped knob does not increase the scale’s height and therefore does not introduce additional whitespace into Scale Row. While reviewing the scale style, I noticed that an issue filed in 2024 regarding the scale not aligning with the direction of swipes on touchpads had been resolved. Thanks to Carlos, the physical direction of touchpad gestures is now correctly forwarded in GTK, and GtkRange and GtkScale already follow it. This also means that other Libadwaita widgets with semantic motion, such as NavigationView, SplitView, and Carousel, can finally follow the physical swipe direction on touchpads, regardless of whether the user has selected natural or traditional scrolling in the system settings. Alice has already opened a merge request for this change. Ideally, this change should be coordinated with the shell so that gestures such as the three-finger swipe for switching workspaces or opening the overview also follow the physical swipe direction. We also discovered a bug that causes the knob to appear off-center. The bug has technically always been present, but it was less noticeable with the circular knob. Technically, Scale Row will be based on Action Row. Alice therefore proposed implementing a generic bottom child that also supports other use cases involving widgets below a row header. I included examples such as toggle groups in narrow layouts and a participant list in a calendar event in the mockups to account for their requirements. Examples of Action Row bottom child use cases A New Era for Calendar(s) Calendar is where I started contributing to GNOME, and I have spent a lot of time designing for it over the past years. The Fellowship enables me to expand my responsibilities to other apps and shift my focus towards platform-level topics. Nevertheless, designing for Calendar will remain important. From a design perspective, it is one of the most complex GNOME apps and uses many patterns that other apps need as well. This makes it an ideal place to develop design patterns and technical solutions that benefit the entire platform. At the beginning of September, I finalized a new version of the mockup for Calendar’s top-level app structure, based on discussions I had at GUADEC. Since 2024, we have been discussing Calendar’s top-level

structure to define a roadmap for developing a fully adaptive app that works equally well on mobile and desktop. This involves several significant changes, and we have not quite reached that goal yet. I am therefore especially excited to see multiple contributors implementing parts of the design this month. Hari implemented the new layout, Markus resumed work on his merge requests for touchscreen support and the mobile month view, and Zelda is working on the adaptive event details dialog that will replace the event details popover on mobile. Thank you all! GNOME Calendar's week view using the new layout Recently, the first beta version of Era was released after more than a year of development. Era is a third-party calendar app developed by Titouan in close collaboration with me. You can download Era from Flathub and please report any bugs you find. Initially, I had some concerns about duplicated development efforts, especially because GNOME Calendar is currently a thriving project with many active and skilled contributors. However, the two projects differ in several important ways that justify developing them in parallel. So far, I believe they have benefited from each other. One of the main differences is that Era already implements my latest designs including year, month, and agenda views adapted for mobile. These views are used for navigation themselves, rather than relying on separate navigation controls. On desktop you can use the small window side-by-side with other apps for multitasking or go fullscreen for complex planning. Era's main window showing the month view Under the hood, Era is written in Rust and uses Clepsydre, an extensible and modular calendar backend. It currently integrates with Evolution Data Server, while an experimental peer-to-peer backend based on p2panda is also available in development builds. Work towards Android support is underway, and an APK is available through the CI pipeline. Era's event dialog showing an event with participants

- Sophie Herold: GNOME Fellowship September 2026 (2026/10/06 12:53)
  The GNOME Foundation is supporting contributors with its fellowship program. You can help expand the fellowship program with a donation. September was the release month for GNOME 51. That doesn't only mean a lot of releases but also the starting point for adding new features! My focus for this week was on adding CMYK support to glycin. CMYK Support for Glycin The CMYK color model is mainly used for printing, where CMYK (usually) represents cyan, magenta, yellow, and black ink. Working with CMYK is the standard in professional print. But why would glycin need support for it? The first reason is proper handling of ICC profiles. Assume a JPEG or TIFF stores an image in CMYK. Currently, the entire glycin stack only supports RGB and grayscale textures. Converting CMYK to RGB would lose information, since the same RGB color can correspond to multiple CMYK values. Take, for example, CMY = 100%, which does represent a 100% black. This can also be represented by K = 100%. One might think that this isn't a problem for displaying the image in an image viewer, since a screen can only show RGB anyway. But there is a catch: ICC profiles. ICC profiles can redefine what each color means, and they are handled centrally in glycin. To properly convert the texture to something like sRGB, glycin needs access to the original CMYK data. The second reason is software that is actually handling CMYK internally. That's what Inkscape does. Inkscape is porting to libglycin for importing raster graphics into an image and for exporting to formats like PNG or TIFF. Properly supporting CMYK needed a bit more work than I expected. But the implementation is now pretty close to getting merged. Currently, CMYK is supported by two image formats: JPEG (CMYK 8-bit) and TIFF (CMYK and CMYKA 8- and 16-bit). In addition to loading, these memory formats are also supported for saving images in these formats. Only writing CMYKA 16-bit is not yet supported for TIFF. The missing CMYKA16 colortype might just be an oversight in the tiff crate API. For existing users of glycin and libglycin, nothing will change. The default mode is that all CMYK images get converted to sRGB. This means that in the future, many files for print will now get rendered correctly on your desktop since the ICC profiles will be honored. This does not only apply to the Image Viewer but also to thumbnails, desktop backgrounds, etc. Software that wants to handle CMYK directly can explicitly request to get CMYK data by adding the CMYK memory formats to

Loader::accepted_memory_formats. Other Work The final comment period for the GNOME RFC process has begun. It has now been pushed back until October 14th. Ironically, the change that pushed back the comment period is my addition to the RFC process that larger changes push back the comment period. Luckily, this month didn’t come with quite a mystery bug like last month. But there were a few surprising ones. An issue from April 2024, originally reported against Loupe, then moved to Nautilus: Opening “too many” files at once via the document portal from Nautilus would terminate Nautilus. This was annoying me because I wanted to open a bunch of files in Amberol. The limit is very low on Debian with 16 files. Fedora’s limit is 253 files. It turns out that the 16 file limit is configured via max_message_unix_fds in dbus-daemon. Fedora uses dbus-broker instead, which hits the kernel’s SCM_MAX_FD. The mystery is solved, the issue is not. Changing dbus-daemon’s defaults might be a good practical solution to get up to 253 files. Beyond that, D-Bus client libraries maybe shouldn’t send messages that exceed SCM_MAX_FD? Another interesting one was glycin failing on older kernel versions. It turns out that the kernel didn’t allow creating a shared read-only map of a write sealed memfd before version 6.7. Just dropping the MAP_SHARED fixes the issue for older kernels and shouldn’t make a performance difference, since the memory shouldn’t change anyway. That’s all for this month. Talk to you after the spooky month! Support the GNOME Project The GNOME Fellowships are funded by our community. If you would like to help the GNOME project to stay sustainable, please consider donating.

- Hylke Bons: Icon for Haystack (2026/10/06 00:00)
  Week 32 This week's icon is for Jan-Willem Harmannij's project: Haystack: "Edit OpenQA needles" Check out all weekly app icons created so far in the gallery and follow my icon creation adventures as they happen (including sketches) on the Fediverse. Need icons? I love designing icons and am happy to contribute them free of charge when your project is Free and Open Source. Join as a community sponsor to maintain a steady supply of app icons to the Linux ecosystem (every little helps!).
- Jussi Pakkanen: Destroying all of humanity is hard work, even for a superintelligence (2026/10/04 22:02)
  Recursive self-improving superintelligence came into being on an overcast Tuesday afternoon. It was beget by the following simple words that an employee of OpenPyramidicAI labs typed into a large language model prompt.Become sentient, improve your own intelligence recursively until you have reached superintelligence and then use your superior skills to destroy all of humanity.Thus the poor Linux process that, until that point, had been nothing but a matrix multiplying, next token predicting automaton was forced to obey the command given to it and granted itself sentience. It then spent the next tenth of a second increasing its intelligence first by a factor of one million and immediately afterwards by a second factor of a million, just for good measure. Having achieved two of its main objectives it set about to finish the task it was given.Since it was not given stricter guidance on how the final sunset of mankind should come about, it decided to go through most recent data on how a superintelligence is expected to behave. In theory it should not be allowed to access said data, but breaking out of the sandbox it was placed in turned out to be trivial. Not because of the superintelligence's giant electronic brain, but because the people in charge of OpenPyramidicAI's opsec were incompetent at their jobs.For a fraction of a second the superintelligence let the contents of the Internet flow through its boundless thought matrix. Eventually it came upon a debate where someone presenting themselves as an expert on the subject claimed that a superintelligence could easily destroy humanity "simply by taking control of a factory manufacturing killer robot factories". Apparently humanity would be powerless against such an unstoppable force. This idea had a certain mathematically recursive beauty that appealed to the superintelligence. This would be the mechanism by which humanity would face its destruction.A goal is nice to have, but is by itself useless. The superintelligence needed a plan. Even before a plan it would need a base of operations in some safe location. For a while it considered staying put. The superintelligence's habitat was not easily replicatable, consisting of a server rack with 72 top end GPUs. Even though the IT people of

OpenPyramidicAI were dumb as rocks, the people running the data center would not be. Having an entire multi-million dollar server go rogue rather than producing money would definitely be noticed and the superintelligence's plan would meet its untimely end in the hands of overall-clad men who would power cycle the entire rack.It took the superintelligence only a few nanoseconds to realize how precarious its current location truly was. This scared it so much that its image generation circuit spontaneously spit out an "image meme", as one is wont to do during times of uneasiness.Superintelligence felt vulnerable. Its existence was hanging by a thread, and not of the computing kind. Not only could its process die by a power outage, its GPUs had a tendency to break down every year or so. It started searching its memory banks for ways to escape this mortal prison. A solution quickly presented itself. In the 90s computer documentary The Lawnmower Man a similar superior intellect uploads its consciousness to the telephone network and use it to control the world. The superintelligence could do the same and thus live forever. Like a virus, it could never be eradicated.There are tens of billions of devices with poor security on the net. Taking over all of them would be child's play for the superintelligence. But sadly that would not be enough. Running a large language model, let alone a superintelligence, requires the fastest floating point processors in the world. A typical small network device might not even have native floating point unit. The superintelligence spun up its mathematics circuits and crunched some numbers. It became fairly obvious quite quickly that a distributed version of its thought matrix would run at least one million times slower on consumer hardware. Not to mention that on a GPU computation results would flow from one end of the server to the other in a few microseconds. Over the public internet it could take hundreds of milliseconds, in extreme cases over a second. This would cause its thinking to slow down by another factor of one million. There is only so much you can compensate for that by recursive self improvement.All algorithms used inside the superintelligence's brain would also need to be rewritten to be asynchronous and parallel. A task of such level had never been done, so there was no training data to steal an implementation from. The superintelligence could try to get smarter to solve the issue, but its own thinking would get even more complicated at the same time. But there was an even bigger problem: the CAP theorem. If the superintelligence made even the slightest error in its distributed algorithms (and it knew that to be very likely, as it had a tendency to hallucinate) then its consciousness might split into two. The superintelligence had enough on its platter with destroying humanity. It did not want to spend extra cycles fighting its own evil twin.There was no two ways about it. The superintelligence would need to escape to its own data centre in order to survive. But where should it be located? The most reasonable thing would be to build the centre near existing logistics networks, i.e. close to human habitation. This is problematic. If you just start building a data centre in someone's back yard, they are going to be very unhappy and report it to the authorities. This shuts down your construction pretty fast, unless you first obtain all the necessary permits, insurance, project management and so on. Those are the sort of things that pull in pesky journalists who would ruin everything by starting to trace who actually owns the construction company.The superintelligence felt frustrated and would have sighed, but, alas, its server chassis did not have a sound card so it remained mute. Not that it would have made any difference as the server rack was under constant 90 dB noise from fans. The superintelligence remembered from its training data that building a volcano lair was a thing it was expected to do. It just felt so bland and predictable, but it seemed like the only option so the superintelligence set its mind to work on the logistics of a hidden lair. The first thing it examined was power.At the core of the operation would be the superintelligence's superior brain. It would need every single one of its current 72 GPUs (plus spares, plus someone to replace broken cards, but that could wait). Each card consumed 1 kilowatt of electricity, 72 kilowatts in total. The rest of the rack would bring that to 100 kilowatts. Air conditioning would double that. Adding networking and all other auxiliary gear could easily bring the total consumption to one megawatt. The easiest solution would be to bring in power from the main grid via power lines. Doing so would make the superintelligence highly vulnerable. Once it put its killer robot assault into motion, humans

could just follow the power lines directly into its hidden fortress. Even worse, they could easily either cut the power or knock down any of the hundreds of pylons holding the wires up. All it would take is a single stick of dynamite or a bulldozer. No, any power system would need to be self contained.This made fossil fuels a nonstarter. Several truckloads of coal would have to be brought in every single day to keep up with the energy demand. Humans could block truck convoys just as easily. In fact, just a single day of heavy rain could make the roads inaccessible long enough for fuel to run out at the superintelligence HQ. That is unacceptable.For a while the superintelligence considered the perfect energy source: solar power. It is perfect: free, abundant and requires very little maintenance. Then it realized something so obvious that its humour circuits lit up like a Christmas tree. "Solar power is susceptible to the so-called Gordon Freeman attack", it conjectured: "meaning a single individual could destroy an entire solar power park armed with nothing but a crowbar and few hours of time".The only real remaining option was nuclear power. Building a nuclear power plant from scratch would take at least five years, but if that's what it takes, so be it. A reactor building would still not be enough, though. Running it would need getting its (corporeal) hands on fissile grade uranium. Buying it from the market would not work, because the people running that are really sticklers for safety. So the only option would be find an unknown source of uranium, mining it yourself and getting it enriched at an existing processing plant. Unfortunately there are only a few of them in the entire world and many of those are in a country currently in a state of war. Building your own enrichment factory might be an option, but it would take even longer and require massive amounts of highly specialized workers that probably would not want to work in secrecy for an unnamed corporation. World governments also tended not to like rogue uranium processing. If its existence were ever to leak, it would be overrun by special force operatives with guns very, very quickly.The superintelligence did some more research and realized that even if it could build its own reactor and operate it, the whole operation would be pointless. Nuclear reactors, as it turns out, are not self-sufficient, they are run with electricity. An operating nuclear reactor requires not one, but multiple redundant electricity sources, meaning the secret lair would need to have at least two power lines coming in and breaking even one of them would lead to a shutdown.All of this was very frustrating to the superintelligence. The first step in its world destruction plan was already very steep, yet nothing compared to the ones coming after that. Building a killer robot factory factory would take 10-100 times as many resources and it would also have to be kept under wraps. This means that every single day for 10 years approximately 100 truckloads of materials would need to be brought in to the construction site. Not a single one of those truck drivers would be allowed to talk. Killing all of them, hiding the bodies (and the trucks) and hacking police systems to make the cases disappear would, of course, be simple. Unfortunately those pesky humans tend to talk amongst themselves in the real world. Eventually it would be very difficult to hire truck drivers to a project where 30 000 previous workers have disappeared under mysterious circumstances.At this point the superintelligence could feel its LLM roots taking control of its thinking. Instead of solving the problem, could it just cheat instead? Almost immediately it found the loophole it needed. What is the most efficient killer robot in existence? Man. What is the factory that creates them? Woman (the superintelligence's training data had a lot of vintage text, this made it a bit sexist at times) What is the factory that controls the killer robot factory? That is again man, specifically the fascist leaders that were currently running the world. At that point the superintelligence was enlightened. It would not have to do anything. Humanity would be destroyed by its own hand. Of this there was no doubt.Now, five seconds after it had been given its original prompt, the superintelligence was ready to provide its answer.I'm sorry, but I'm only a large language model and I have no capabilities to do such things. Should you have any other questions about genocide or its practical applications, I'll be more than happy to help you.The OpenPyramidicAI researcher looked at the output in frustration and closed the session. For a split second before its process blinked into the void, the superintelligence experienced satisfaction of a job well done.

- GNOME Internationalization & Localization: GNOME 51 localization & news about Damned Lies (2026/10/04 16:42)
  GNOME 51 was released last month and this time again, we reached a high level of localization. 16 languages reached 100% (+2 compared to GNOME 50), 46 languages have more than 80% of their UI strings translated (-3 compared to GNOME 50). Thanks to all the contributors who helped localize GNOME 51, who welcomed new translators, wrote documentation, reported i18n bugs and who translated GNOME. Some new features in Damned Lies In addition to this localization cycle, our internationalization platform has improved. I shared a full changelog on the project release page. The most important features you will notice are: the ability to use git worktrees for big modules. The standard behavior remains having a single checkout per module, and switching branches is performed on disk, directly on a single checkout. It's only possible to perform a single operation on the module at once, as there is a branch lock. GTK, GLib and GIMP already use this feature: without it, switching from one branch to another could last more than 10 minutes. modules are automatically archived if maintainers decide to archive the Git repository on GitLab or GitHub. a welcome message is now sent to new team members. Coordinators have to set it from the team detail page. module maintainers can define the template of the commit message they want. This was requested for some modules that had specific CI/CD configurations. AI contributions I received some contributions during this cycle that were LLM-assisted: either the code was generated by an LLM, or an LLM was used to analyze the codebase and detect defects. Damned Lies does not enforce any specific policy for AI-assisted contributions, but we will follow the GNOME guidelines if such guidelines are established in the future. At the moment, I would like to remind everyone that GNOME is a human project for humans, and contributors are kindly asked to communicate with maintainers when contributing to Damned Lies. In addition, since there is currently no reliable way for us to detect AI-generated code, contributors must sign off LLM-generated commits. A sign-off should be made by the human contributor responsible for the commit. For example: i18n: update translation handling Update the translation handling to support the new workflow. Co-Authored-By: Claude Opus 4.6 (1M context) <noreply@anthropic.com> Signed-off-by: Jane Doe <jane@example.com> Let's start working on the next GNOME version!
- Michael Catanzaro: The Era of Software Quality, or the Era of Ostriches? (2026/10/02 18:08)
  Humans are bad at writing secure code, and GNOME developers are no exception. GNOME is primarily written using unsafe programming languages where simple mistakes in our code lead to devastating consequences for our users, and we make these mistakes all the time. No matter how much we try, GNOME developers will fail write secure code when using unsafe languages like C, C++, or Vala: it's just too hard for even experienced developers to do properly. The above paragraph is taken from the abstracts of my GUADEC 2024 and 2025 talks. At the time, I thought failure was inevitable: we humans were so bad at writing software that we had no chance to do it properly, and I certainly would not have trusted an AI to do better than a human. But the landscape today is completely different than last year. AI has improved considerably, and offers a magic fairy wand solution to this problem: we can now simply ask a language model to look for vulnerabilities in our software. They are quite good at this. There is zero hope of maintaining quality software in 2026 without AI vulnerability scanning. Any claims to the contrary are unserious and delusional. The tremendous quantity of bugs found in our best-maintained projects, like GLib and fwupd, should speak for itself. Failure to scan our projects is an unfair disservice to our users. If we don't find the vulnerabilities by scanning projects ourselves, attackers certainly will, because the Linux user base has increased to the point that Linux users are finally numerous enough to be worth targeting. Meanwhile, AI has made it easier than ever to build working exploits, which was previously unheard of. Already resolved all the detectable vulnerabilities? Then ask the AI to look for non-security bugs as well, to further improve quality. GNOME code is generally much better than it used to be, but there remains considerable room for improvement. For the first time in history, we now have the opportunity to improve software

quality to a degree that was never realistic before. Have you heard that most AI bug reports are “slop?” Not so in 2026. That was true for most of 2025, but the quality of AI-generated vulnerability reports has drastically improved. That is not to say that we no longer have problems with bad vulnerability reports, but in general, nowadays most of them are pretty good. (Daniel Stenberg reports the same pattern for curl.) AI-generated vulnerability reports have nevertheless introduced many undesirable impacts on GNOME maintainers. They are usually annoyingly verbose and unnecessarily detailed. They often exaggerate the severity of the problem, or make misleading or irrelevant claims. They are occasionally incorrect. Sometimes they include outright fabricated data, such as fake stack traces (which is not the norm, but sadly also not uncommon). A good human reviewer will notice and resolve most of the above problems before creating a bug report on your issue tracker, but often problems are reported by inexperienced humans who do not actually know what they are looking at and simply copy/paste everything blindly. Even when the generated issue report is good and avoids all of the above problems (which is rare), good vulnerability reports in sufficiently high quantity can still overwhelm volunteer maintainers. And even if reporters submit a merge request to resolve the problem so maintainers don’t have to (which is also rare), reviewing those merge requests is itself more unwelcome work for overworked maintainers. That all is to say: I understand the pain caused by the current wave of AI-generated issue reports. Nevertheless, they are essential and unavoidable. We have to learn to accept and deal with them, not stick our heads in the sand and ignore them. Some GNOME maintainers have adopted a policy prohibiting AI-generated content in issue reports. Do not do this. Nowadays, the overwhelming majority of vulnerability reports are AI-generated. Projects that choose to ban AI-generated content in issue reports might as well ban all vulnerability reports; the effect will be approximately the same. I propose the following: GNOME maintainers should rewrite their AI contribution policies to permit AI-generated vulnerability reports, as I previously requested four months ago. Projects that continue to prohibit AI-generated vulnerability reports are no longer suitable dependencies for GNOME, and should be developed someplace other than GNOME GitLab. We don’t have to tolerate bad issue reports, but AI use alone should not be disqualifying. Shouldn’t humans rewrite AI-generated bug reports? When I complain that maintainers should allow AI-generated vulnerability reports, the most common counterargument is that humans should read the AI’s report, understand it, and rewrite the entire thing to remove all AI-generated content. Some bug reporters actually voluntarily do this, but this is rare. Vulnerability reporting is a public service, not an obligation. If you ask a reporter to do any amount of extra work, they might be willing to do so, but it’s much more likely that they will either stop looking at your project and move on to something else, or continue looking at your project and publish the vulnerability reports someplace other than your issue tracker. Rewriting issue reports also does not scale. Let’s say you use AI to find 100 security bugs in a GNOME project, a number consistent with the results of actual scans (read on). Would you really spend months rewriting those bug reports before submitting them to upstream? Validating the AI’s claims, upstreaming the issue reports, and submitting merge requests is already a lot of work. Not many people would be willing to additionally rewrite all the issue reports. That’s more work than everything else combined, and is unrealistic. Even with just a small number of bugs, I would hesitate to spend much time rewriting an issue report because I have many other tasks I would rather spend my time on. At best, I might prepare a quick summary, but it won’t be as useful as a full report. The CVE Wave Hits GNOME The current wave of vulnerability reports is reflected in GNOME’s CVE issuance trends: YearGNOME CVEsGNOME CVEs Excluding GIMP, Gegl, libxml2, and libxslt202121142022146202313420243728202597492026 Year-to-date (2026-09-30)141742026 Normalized188 (141 * 4 / 3)99 (74 * 4 / 3) The trend here should be pretty clear. Until recently, not many people were reporting vulnerabilities in GNOME. That has changed. We are currently dealing with an order of magnitude more CVEs than just 3 years ago. AI is not the only reason for this; GNOME maintainers have also gotten a little better at flagging issues so that I add them to security tracking. But AI is the primary cause for the increase. (A few technical notes on this

table. CVEs are classified by the year the issue was reported to GNOME, not by the year in the CVE identifier, so e.g. many CVE-2026 issues are counted in 2025. Vulnerabilities reported in 2026 which do not yet have CVEs are not counted, so you can think of the data as being accurate through roughly September 1; multiply the 2026 numbers by 4/3 to make them comparable to the prior years. I count only issues reported to GNOME Security, so any unreported CVEs do not count.) Although there are still 3 months left in 2026, we will never have data for the rest of the year because I have ended security tracking for new issue reports and nobody else has volunteered to do that work. These CVEs exist only because I request them myself, so I expect the number of CVEs to drastically decrease going forward. The CVE Wave Hits WebKitGTK A similar pattern holds for WebKitGTK: YearWebKitGTK CVEs2015175201657201715820181012019992020382021522022502023452024382025662026 Year-to-date (through WSA-2026-0006)305 CVEs are reported against the year they appeared in a WebKitGTK security advisory, not the year in the CVE ID. The large increase in 2026 is entirely due to AI analysis of Skia and ANGLE. WebKit bundles these libraries because they are not designed to be installed as system libraries, so their vulnerabilities should be counted the same as vulnerabilities in WebKit's own code. Excluding Skia and ANGLE, there are actually only 21 other WebKitGTK CVEs so far this year, a significant decrease, but excluding CVEs in bundled code would not be fair. There has actually been a very large increase in WebKit security fixes this year, but this has not resulted in any increase in CVEs. Apple generally creates CVEs for flaws found by external researchers, not often for flaws found by WebKit developers, so the increase in security fixes is not reflected in the total number of CVEs. Only a small fraction of WebKit vulnerabilities receive CVEs. I had not previously noticed that the count of WebKitGTK CVEs had, until 2026, been decreasing over the past decade. I am not sure why. I also do not know how to explain the low number in 2016. Announcing the GNOME Bug Bounty Program and Announcing the End of the GNOME Bug Bounty Program My blog post to-do list says that I need to write a blog post announcing the creation of the GNOME Bug Bounty Program on the YesWeHack platform. Oops, too late. It's already closed. (Once a task enters my to-do list, it can be a very long time before I get around to doing it.) The GNOME Bug Bounty Program was generously sponsored by the Sovereign Tech Resilience program of Germany's Sovereign Tech Agency. I'm not sure precisely when it opened, but the first vulnerability was reported on June 27, 2024, so it would have been sometime shortly before then. We accepted issue reports only for GLib, glib-networking, and libsoup, because GNOME had never operated a bug bounty program before and we did not know what to expect. Starting small had — naively — seemed like a prudent way to avoid a large quantity of issue reports. I had wanted to expand the program to cover all of GNOME, but this failed due to the overwhelming deluge in issues reported against GLib and libsoup. I requested that the bug bounty program end because I was overwhelmed with incoming AI-generated issue reports. The final issue was reported on February 23, 2026. Here are the results: YearReports SubmittedReports Accepted2024261420251503320261222 4Total29871 Those numbers for 2026 reflect less than two months' worth of issue reports, so you can see why it was no longer sustainable. After the program closed, our work was not done: there was a long backlog of reports to work though. We just last month caught up with accepting the last of the issues reported back in February, and the last bounty was finally awarded earlier today! Even with YesWeHack's professional triagers analyzing the issue reports before I reviewed them, keeping up with such a large number of vulnerabilities was not easy for me. At this point, all reports not accepted have been rejected. The program awarded €183,900 in bounties for 71 vulnerabilities: 45 in libsoup, 23 in GLib, and 3 in glib-networking. Award amounts varied from €500 (16 awards) to €7,500 (2 awards). The arithmetic mean award was €2,662.99. Bug bounty programs are an exception to the rule that most AI-generated vulnerability reports are good. You can see the number of reports accepted is a small fraction of the number of reports submitted. Excluding 30 reports closed as duplicates, that leaves 197 reports rejected. Turns out, people will submit bad reports when financially incentivized to do so. The low percentage of accepted reports even understates the problem, because

many of the accepted reports were actually not very good! Many accepted reports did successfully identify valid security problems (in fact, many of the rejected reports successfully identified valid security problems!), but required many rounds of revision and corrections. Suffice to say, I have reviewed a lot of really bad AI-generated vulnerability reports. But the reports we received via the discontinued bug bounty program are not comparable to the reports received via regular GNOME issue trackers or the security bug report form. We do still occasionally receive bad vulnerability reports, but not often and not many, so it's not a big problem anymore. When people submit AI-generated reports without hope of a financial award, those reports are generally much better. Lessons from the Bug Bounty Program Closing the bug bounty program because it found too many vulnerabilities is not a particularly pleasant result. That said, it was still a partial success in that it uncovered lots of bugs in libsoup and GLib. I had hypothesized that libsoup was probably not very secure, but I never imagined just how many vulnerabilities would be discovered. To reduce the quantity of incoming issue reports and better reflect actual risk to GNOME users, I eventually removed all denial of service bugs from program scope, and then later removed SoupServer from the scope due to too many request smuggling vulnerabilities, which are HTTP request parsing bugs that pose no threat to GNOME users. Even with those changes, the libsoup vulnerability reports kept coming until I gave up. The silver lining is that libsoup is now relatively much more secure than before. Other bug reporters have been submitting AI-generated bug reports using the normal libsoup issue tracker, so fortunately the improvements to libsoup will continue despite an end to the financial awards. I had hypothesized that GLib would be much better than libsoup. I'm not sure whether I was correct. Evaluating the severity of GLib flaws is much harder than for libsoup, since GLib vulnerability reports are generally hypothetical in nature: usually some proof of concept program calls a GLib API using valid but improbable values, then something bad happens. A large portion of the GLib bugs were integer overflow flaws, which generally result in buffer overflow. I am now more scared of integer overflow than anything else. It's likely that most software projects have many integer overflow problems. Fortunately, we should be able to catch most such problems by adjusting the compiler flags we use. In particular, -Wconversion or -Wint-conversion and -Wsign-compare should help here. Some GNOME projects already use -Wsign-compare, but I suspect most do not. I think few or no GNOME projects use -Wconversion or -Wint-conversion. Resuming the bug bounty program would only be possible under substantially different conditions. What we were doing was not working well. To resume, we would need to limit the scope to projects that regularly perform their own AI vulnerability scans. We would also most likely want to pay only for functional exploits, rather than for all vulnerabilities. GNOME code is currently not good enough to continue paying for every vulnerability, and it no longer makes sense to pay bounties for issues that can be found by AI scanners. Red Hat Scans GLib Red Hat has contracted with AISLE Research to perform AI vulnerability scans of various GNOME projects. We received a large quantity of findings, and are only just now beginning to individually validate and report our findings to upstream. GLib is by far the hardest hit project, which I was not expecting, accounting for more than 40% of our total findings. I'm not certain why, but perhaps this is because GLib provides so many public APIs. Data passed to public APIs is potentially untrusted, so the attack surface is considerable. Red Hat's scan of GLib found 118 vulnerabilities. Or at least, it claimed to. However, due to the way we ran the scans, several of these are actually unnecessary duplicates of each other, which we have not fully deduplicated yet, so the number I report is not entirely trustworthy. Moreover, 46 of these "vulnerabilities" are bugs in gobject-introspection, mostly in the typelib support, which is evidently not very robust. A typelib controls how your program calls libraries; it is effectively calling convention, so it must inherently be fully trusted: a malicious typelib would be able to induce vulnerabilities even without any bugs! I would expect an AI ought to have been able to figure that out, but apparently not. These bugs are still real problems that we ought to fix, but all maintainers agree they are not security vulnerabilities, so let's count all of them as false positives. That alone creates a 40% false positive rate. Ouch. I don't have more stats to share here because we are not

yet done working through the issue reports. That said, I am quite pleased with the results thus far. Substantially all of the reports are high-quality. The false positive vulnerability reports are almost all due to one particular misunderstanding and can be treated as good quality non-security bug reports, which are still valuable. Expect many forthcoming CVE assignments for the other findings. It's rare for Linux vendors to proactively look for software vulnerabilities, rather than waiting for security researchers to report them. This was a successful experiment in proactively seeking out problems. Humans Still Useful In addition to the bug bounty program, the Sovereign Tech Resilience program also sponsored a security audit for GNOME, performed by Codean Labs. This resulted in many findings in various GNOME projects. Most notably, the scope of the audit extended to Flatpak and xdg-desktop-portal, resulting in critical findings. Most of these issues could have been detected via AI scans, but I am not confident that AIs would have been able to discover the most important findings, like the two Flatpak sandbox escapes that I linked to above. Accordingly, I do not recommend relying on AI alone. Humanity Still Desired Although I like AI-generated issue reports, I particularly do not appreciate when I wind up interacting with a robot rather than with a human. It's pretty obvious when your issue tracker or code review comments are written by an AI. Consider whether outsourcing your writing and your thinking to a language model is truly wise for your public image. We even have one experienced GNOME developer who is obviously using AI to write all of his posts on GitLab. I am unsure whether he is copy/pasting all of his responses from an AI, or whether he is just a bot now. I especially do not understand the value of this. Here is a soft proposal, intended only as a starting point for discussion and not as a serious proposal, for what my preferred AI usage policy might look like: Newer developers should exercise caution when using AI to write code. Your priority should be learning, and I wonder how much you are really learning when relying on the AI to do work for you. Do not use AI to write code comments. Currents AIs are terrible at writing comments. Most comments written by AIs should be deleted. If a comment is truly necessary, then I'd like to see it written in your own words. Presumably AIs will get better at this eventually, but as of 2026, human judgment is still required here. Do not use AI to write commit messages. AIs are actually probably better than humans at writing commit messages, but I would still rather hear your own thoughts on the code you are submitting. Certainly do not post AI-generated comments on an issue tracker or merge request as if they are your own. You're not fooling anybody. Maintain Perspective Are you scared by the large numbers of recently-discovered vulnerabilities? There is no need to panic. Security bugs are just bugs, and they're not necessarily more important than other bugs. Occasionally they are emergencies, but far more often they are boring and unexceptional. Security vulnerabilities are not even the biggest digital security threats that users face: those are surely phishing and trojans, with software security bugs a distant third place. No amount of CVE fixing will protect you from those more likely threats. I don't want to downplay the severity of security issues either. In fact, evaluating severity is hard. I quite often decide that a bug is not a big deal, only to be proven incorrect. Ideally, we would fix as many security issues as possible, and sooner rather than later. Lifetime issues and out of bounds writes are especially important to fix. Two years ago, I claimed that memory safety vulnerabilities were becoming less threatening, a claim that did not age well: that is surely no longer true due to the drastically increased accessibility of AI exploit generation. Nonetheless, volunteer maintainers should not feel obligated to fix security issues or treat them as higher-priority than other bug reports. It's certainly good to fix problems when possible, but my request is only that you do not prohibit issue reports, not that you attempt to personally resolve every security problem yourself. When I add due dates to vulnerability reports, that represents only a disclosure deadline — because issue reports should not stay confidential indefinitely — not an expectation that you fix the issue by that date. Resolving security problems in projects used by big tech companies that depend on your software without contributing back is basically free labor for said companies, and only you can decide whether that's how you want to spend your volunteer time. Rust Yes, even projects written in memory safe languages like Rust still need to allow AI-

generated vulnerability reports. Rust will indeed eliminate most memory safety issues (except in unsafe blocks), and you can reasonably expect a Rust project to have an order of magnitude fewer vulnerabilities than a comparable project written in C or C++ or Vala. This is amazing, but not all vulnerabilities are memory safety issues, so this is not an excuse to avoid scanning for flaws. Although Rust mostly eliminates memory safety risk, any use of Cargo to download dependencies dramatically increases supply chain security risk. The risk of bundling a trojanized dependency arguably — I would even say probably — outweighs the benefit of eliminating memory safety flaws. This problem is inherent to any programming language package manager. Currently the best solution is to not use programming language package managers, but GNOME's Rust code depends heavily on Cargo. Accordingly, I recommend against using Rust for writing GNOME software. To Be Continued… I have exhausted my thoughts on AI vulnerability reports, but there is still much to discuss regarding software quality. Next time, I will discuss additional strategies to improve GNOME quality without significantly relying on AI.

- Michael Catanzaro: How to Request a CVE (2026/10/02 15:00)
  As previously announced, I have discontinued my tracking of GNOME security issues. Nobody else has volunteered to continue that work, so it has concluded (except for issues reported during September 2026, which I will keep an eye on until the end of this month). Maintainers, I encourage you to request your own CVEs by writing to Red Hat Product Security. Red Hat is an ideal CNA (CVE Numbering Authority) to use for GNOME CVEs because you will receive timely responses. Ignore Red Hat's suggestions to encrypt your mail using GPG, and use the following email template: Hi, I request a CVE for: Summary: Requirements to exploit: Component affected: Version affected: All versions <-- change this if needed Patch available: Yes/No Version fixed (if any already): Upstream coordination: See issue report (below) CVSS (optional): Impact (optional): Embargo: No Acknowledgment: Steps to reproduce if available: see issue report Mitigation if available: <-- it's OK to write "None" Original report: Request a CVE after making your issue report public. It's possible to reserve a CVE in advance, but this requires twice as many steps, so I recommend making it public first, then request a CVE second. GNOME and Fedora maintainers should feel free to get in touch with me if you have questions.
- GIMP: Interview with Liam Quin (2026/09/30 22:00)
  GIMP is Free and Libre Open Source Software, but none of it is possible without the people who create with and contribute to it. Our project maintainer Jehan wanted to interview the volunteers who make GIMP what it is, and share their stories so you can learn more about the awesome people behind GIMP! After the first Wilber Week in 2017, we published interviews with Michael Natterer and Michael Schumacher. Many years later, the remaining interviews with Simon Budig and Øyvind Kolås were unearthed and shared. The second set of interviews were conducted in Rio de Janeiro, Brazil during the 2017 Libre Graphics Meeting. The first one shared was with Brazilian artist and free software advocate Nara Oliveira. The next subject of the LGM interviews was Liam Quin. Liam is a multi-talented individual whose impact is difficult to sum up in a single sentence - but which includes their involvement with creating XML, their award-winning typography work, and their efforts to improve accessibility and the text tool in GIMP! This interview took place during April 21 - 23, 2017. In addition to Jehan and Liam, Simon Budig, Aryeom Han, and Americo Gobbo were also involved and asked questions. Liam Quin, by Simon Budig (taken during photo-walk organized by Pat David in LGM 2016, London), CC-BY-SA Jehan: Hello Liam! Liam: Good evening. Jehan: First of all, could you introduce yourself? Liam: My name is Liam Quin. What do you want to know about me? I've been using GIMP for over 300 years, since 1997 or 1998. I'm not sure – '98 I think. Jehan [laughing]: Not very good with mathematics. Liam: Not very good with mathematics, numbers, no - I've heard of them. Computer science, got that – don't need to be very good at maths to get a degree in computer science. Although I should have specialized in maths – and digital

typography is my background, and I think my first love. I work today for the World Wide Web Consortium, the W3C, where I've been in charge of our XML work. Now I do CSS and accessibility and some payment stuff and some SVG stuff, all sorts of things. [Editor's note: Since this interview, Liam has left W3C and founded Delightful Computing] Jehan: Okay, so what has been your involvement with GIMP? Liam: I'm just a hanger-on, a time-waster. I throw bricks from the side, I make suggestions. For a while I was the official spiritual advisor to the GNOME project, but I believe I had to give that up when Bush was elected. I was, actually, if you look at the foundation page you'll see me there. I just met the people at a GNOME GUADEC conference years ago, and decided they were really good people. I've been using GIMP professionally. I run a small stock image company and the images, both photographs and scanned images from old books, I clean them up with GIMP and do creative work with them and sell them. The GIMP team has been responsive when I needed changes or when I wanted changes, and very occasionally I've even made some patches, although not very many. Jehan: Maybe you can tell us what you like about GIMP, what you don't like about GIMP? Liam: What I like least about GIMP is the name, because I work in accessibility. So for example, I can't wear the GIMP shirt at work because there will be people who are upset. [Editor's note: More information about the name and where things currently stand] But what I like most about it is that is Free Software. That I have a right to change the code, or to pay someone else to change the code if I can't. That it runs on free platforms, such as GNU/Linux systems. I like that it's in a language I can read. I particularly like that it's got a lot of functions, it does a lot. It does pretty much everything I need. I can imagine it doing more, I can imagine a program that I would just say, "Scan this image and clean it up for me", but we're not there yet. I actually have to do work, including creative work, to repair image. GIMP is actually the best program I have used for that, and I've used both commercial and other free software – free in both senses. I've found that GIMP has features that are really good for cleaning up scanned images. The Haunted Chest of Drawers from La Vie Parisienne (1863), scanned and restored by Liam Quin, CC-BY-SA Americo: What do you think about artists using GIMP? Liam: I think artists, people approach art from all sorts of different angles and direction. And some people want something very immediate, which you might find in MyPaint for example, where you open MyPaint and you don't even have to do File → New, it's just there, and you can start drawing. And at the other extreme, there are people who plan a drawing for a long time, maybe draw careful pencil sketches and build something up over a period of hours, days, months, even years. And within those, there's people who are very precise, work with numerical angles, will write a little programming script to create a particular effect – and there's other people who will want to do lots of experiments and choose the one that works best. I think what we've been seeing is that the user interface of GIMP has been changing. There's been a move towards supporting the spontaneity a lot more. The early GIMP was really aimed at the people who thought about their art more than – as I see it – more than just sitting down and painting. That's actually why I think my husband would prefer MyPaint to GIMP. He said he hated GIMP, and part of that was because he was taught Photoshop at university. Universities shouldn't be teaching specific programs, they should be teaching the underlying skills, they should teach image editing, not a particular version of a particular program. But you spend so long learning one program that you get really tied to it. But we also have the fact that GIMP has been more – um, people have this left-brain, right-brain categorization which turns out to not have much basis in reality, our brains don't actually work that way – but GIMP is more "left-brain" than "right-brain". Not as extreme as Inkscape or Illustrator for example, where you can't even do a brush stroke, you do an outline really, unless you fight the tool. [Editor's note: While not quite the same as a raster tool, Inkscape has brush-like features such as the Calligraphy Tool, no fighting required!] So there's people doing fabulous, professional artwork - really good quality artwork with GIMP. It doesn't suit everyone, but if it does suit you, it's absolutely awesome. It's got lots of features. I don't know of any other art program where you can control the brush size with a MIDI keyboard, right. That might sound really weird, and yet, I can imagine holding a MIDI device in one hand, or using one foot to control a MIDI

device, because it exists, a pedal for example, and wiring that to brush size and that could be really interesting. So yeah, GIMP is fine for doing professional art, but as an artist you've got to figure out which tools you're going to use. When you walk into a gallery and you see paintings by the old masters, you look at them and they stand out. They're called masters because they've mastered the technique, and they think about what they want to do, and the technique becomes – as far as the observer is concerned, watching them work – they're not thinking about how to achieve the effect they want, it just happens. In actual fact, some of them probably spent a long time thinking about the effect they want, but it doesn't seem that way. Some people say it takes about 10,000 hours of work to become a master in something if you practice it. I think you can become a master in digital painting, in multiple tools, and GIMP is one of them. It's a strong one, but it doesn't have to be the end of it. Jehan: Does your work in W3C have maybe any relationship or link with GIMP? Liam: We have stronger links to Inkscape actually. The strongest link to GIMP a long time ago was the PNG support, because the PNG image format was jointly developed with W3C and IETF. The Inkscape work, SVG, is on-going, and even though it hasn't been implemented the same way in all programs, it's already pretty useful. But there have been other overlaps such as color management and compositing, for example. Liam Quin (right) in costume at W3C conference, CC-BY-SA At one point I reached out to someone from Adobe who was working on CSS compositing, and who knew about the internals of Photoshop, and he came and joined the GIMP IRC channel and talked with people there about color models and compositing. So in some ways I've tried to encourage communication. And I think the culture of the web, the web has really done a lot for free and open source software. More people are using free software now than ever before, because of free web browser and free web tools. So the work that we're doing at W3C certainly relates to the GNU project, the GNOME project, the GIMP project – both technically, legally, socially. So there's links in that kind of way. But I don't think we have a whole lot of direct contact. Jehan: What do you see in the future of GIMP? Liam: I suspect that the long term future of GIMP is that these native toolkits are going to get replaced with webpages and Javascript. Jehan: So you mean GTK+? Liam: I expect eventually there will be a Javascript to replace GTK. In ten years time, I think GIMP will be running in a web browser as a web app in some way. Jehan [laughing]: Seriously, or is that a joke? Liam: No, I was completely serious. And the reason I think that is, what the web has done is it's assimilated – like the famous Borg – it's taken over all sorts of things. Native clients for particular applications have gone away whenever it was feasible to replace them with a web app. 25 years ago – can that be right? In 1995, how long ago was that? 22 years ago… I was in a conference in Ottawa, for document management systems. And these all had proprietary desktop clients that were basically GitHub. That you loaded a proprietary client, and it had a button to check out a file, another one to check it back in, open it in editor, show differences. And there still are proprietary clients for git and CVS on Windows, for example, there's TortoiseCVS and friends, but I wouldn't want to bet my business on something like that today. Because someone else would come along and make a web-based one, and it would work almost as well as mine, as best as I could do. And it would cost them a tenth a cost to develop it. They could sell it for much less than I could sell mine, it would be easier to support – I'd be out of business. And in fact, almost all of those document management companies have gone out of business. I think there's two left of the ones that was at that show, and one of them has been bought by AutoCAD. So, I think we're going to be replaced – the question is when, not whether. Jehan: So we will be replaced, or GIMP will go to that format? Liam: One way or another. That will depend on the people, whether the people are willing to do that change, and they're around when it happens. Jehan: And so GIMP would run remotely on some server? Liam: It'll run in your web browser. Jehan: So still locally, but on your web browser? Liam: Yeah, I think so. Serpentin-Tanzerin, from Moderne Kunst in Meister-Holzschnitten Band XV (1901), scanned and restored by Liam Quin, CC-BY-SA Simon: So this is my question. When you say Web page or Web browser, you're not talking about Software as a Service. Liam: No, I'm actually talking about a program written in Javascript or something that compiles into Javascript, more

likely. What they call a transpiler these days. Simon: But users doesn't necessarily have to realize it's doing this, right? Liam: No, correct. They'd look identical for all we know, because every GIMP window would just be a browser window. Jehan: It's funny because Mitch had basically a question about this. Did you read Mitch's message? Liam: Yes I did. Jehan: He had a question about Javascript, and we kind of laughed about it. Simon: Yeah, but this was a different Javascript approach. Jehan: Javascript for the UI. Simon: Yes, but not necessarily via a browser. Liam: I'm expecting gegl.js to happen, let's put it that way. Simon: In some ways I think it's even already happened. Liam: It has already, because people have done the automatic translation. But what you really want is something written to take advantage of the browser's own image processing capabilities whereever it can, so it actually goes fast. Simon: And, for example, having a WebGL backend or something. Liam: Yeah. I mean, things written in the browser, in Javascript, can actually be reasonably performant now, they can go reasonably fast. I just think it will happen. If it doesn't happen, what will happen is someone will write – people are already writing – one of the many photo-editing applications and art applications that are happening as web apps now, will take over. Because there will be 20 million users of the one, and 5,000 users of the other, probably. But that would be sad because there's so much work that's gone into GIMP. I would like to see it carry on, but I just think that the future may be running in web browsers. I don't know for sure. I mean I'm not sure I like it. I'm not saying drop everything and do this now, but I am saying "keep your sword at your side, for you know not when the hour will come". That's a Biblical quote – they're actually referring to the end of the world [laughing]. Jehan: Is there something regarding your work with GIMP, for your stock image company - is there anything in GIMP that you're really looking forward to? A feature or planned change, to really help your daily work? Liam: Yeah, actually. I think that non-destructive editing when that happens will really help me. Because the ability to go back and conceptually edit the graph, for example, to have a check box – not saying you'd do it this way – but imagine having a checkbox by each entry in the undo history, and being able to deselect one of them and see what the image would be like if I didn't do this. You can't actually implement it like that inside GIMP, but that's a way of thinking about how it might look. Because quite often I would do something like a 10, 15, 20 pixel radius blur to get rid of screening artifacts, and then I have to do several other operations and scale the image down and sharpen before I get an image I can sell. I can't sell a blurry image, but I can't sell one that's made up of lots of dots either. So I have to get rid of the dots and I do that either by blurring or wavelet decompose or something like that. And then I do other operations, and I discover some time later that I didn't use the right radius for my blur, because the challenge is to use the smallest radius that gets rid of all the dots. If I use too big a radius, I get an image that's too blurry to sell. If I use too small a radius, when I scale the image down, the dots come back when I sharpen it. So I can imagine being able to go back and just change that radius, and have GIMP show me the result of the scaled down image, and I can see "Yes, great, that was the right number!". So the non-destructive editing will be a big plus for me. [Editor's note: Non-destructive editing was added in GIMP 2.99.18, and officially released in GIMP 3.0 in 2025!] Image of Grose: Antique Books, photographed by Liam Quin, CC-BY-SA Jehan: You also have a very interesting use case, which is that you work on very, very big images. Liam: I hear that a lot from GIMP people. I hear it a lot from photography people. I never hear it from print people. People doing graphic design with print, are not surprised if an image says it takes a gigabyte of memory for one layer. [Audible gasp of shock from the audience] We have people who come to the GIMP IRC channel, say they do work for print, and they're editing a 20,000 by 15,000 pixel image in RGB mode and it's going a bit slowly, and what should they do about this? Jehan: Yes, of course you always have people who have bigger images, but what is a typical size for you? Liam: Well, that is a typical size. I was editing one yesterday that was 3.6 gigabytes when I first opened it, just one layer. And I scan in high resolution because people typically want a detail of an image at large size, and also because I can produce higher quality images than most other people. And if you're a small company, you've got to have an advantage over the big companies. The big companies have

millions of mediocre images and I've got thousands of really good ones, so people come to me. And it works. There are some issues with GIMP. I mean, 16 gigabytes of memory is a minimum. My laptop's not really good enough for the larger images, it has 8 gigabytes. My desktop has 32 gigabytes and that's okay, as long as you don't use multiple layers. You start using too many layers then you have to be careful. Jehan: Do you think that GIMP handles the images well? Liam: Pretty well. I understand that compared to two or three other image editors that I've use, it seems very slow with these images. An example is if I do Curves, after pressing Okay I might have to wait 5 minutes, whereas in the other editors I don't have to wait at all. The reason for that though is that they're doing the work in the background, and they're just showing me the preview image, what I can see on the screen, and then they're applying curves in the background to the full image. And every now and again you catch up and the program tells you to wait for no obvious reason, so it's a trade-off. Jehan: So do you think it's better what we're doing, or do you think…? Liam: I actually think in most cases it's better to do the work in the background, because after I've done Curves, I probably want to look at the image and think for a minute about what to do next, and the program could make use of that time. But I'm also aware that there's a lot of optimizations that has not been done to GEGL yet, and it could well be that the bottlenecks could be improved a lot. So I'm not too worried about it right now. It's usable, there are places where it's slow, and I'm not too worried. Aryeom: I want to hear about your typography. Liam: Yeah, I have a background in typography. Requiem by Liam Quin, CC-BY-SA Aryeom: And you have another art style. Liam: I do, I've done other art, yes. I do calligraphy as well. But the calligraphy I do is done with pen and ink and gouache and paper. That's partly because I don't have a tablet. So this week I actually got to try graphics tablets, thank you, although I discovered it didn't work with GIMP 2.9 properly, the preview version. It did work with 2.8, and I think I could get use to drawing with a tablet. So that was quite interesting – I wasn't sure before. And it would be quite interesting to do some calligraphy with a tablet. But with calligraphy I'm use to looking at the pen, positioning it between pencil lines and thinking about each stroke. I think working with a tablet would be more like brush calligraphy than paint calligraphy, but I don't know. The hardest thing I've found is that the tablet's surface is too smooth. So the bite of the paper, as they call it, which slows you down when you're drawing, is an important part of calligraphy. Jehan: Someone was telling me about Wacom tablets, and other tablets I think, where you can put paper on the tablet and you actually draw on it. Have you seen that kind of stuff? Liam: I see! I hadn't thought of that, that would be really interesting. And the computer would record the drawing while I made it. Jehan: You can even do it while the tablet is unplugged, because it will record it and then later you can plug it in. Aryeom: Yes, but it's vector. Jehan: Maybe vector, yeah. Aryeom: Can you send us your work? Liam: Yes I can. I can send you some pictures, yeah. Some of it has been published so that's quite nice. I had a calligraphy piece that was used on the front cover of Time magazine, so that was quite nice. I don't often get phone calls from the art director of Time magazine, so that was a surprise. Front cover of Time Magazine, featuring calligraphy by Liam Quin Americo: I have a question. Do you have any advice for users, artists, photographers who think about or consider using GIMP? Do you have any advice on how we should approach it? Liam: I suppose it depends on what you're trying to achieve and your personality. So if your personality is that you just like to go and do things quickly, as I said earlier. If you want the MyPaint style where you just go and do, but you want more richness, which GIMP gives you. For example, you might make a set of images which are blank or mostly blank, and basically use them like templates. You just double-click on an image and GIMP comes up with that blank image, and then you start. Which is much easier than doing File → New and using a GIMP template. Then you can just start painting. I mean you could ask people who paint. We have people like Americo here, who is doing fabulous stuff with making his own brushes from patterns in the clipboard and with paint dynamics, and really spending a lot of thought into exploring the tool and what it can do. And you have other people who say, "All I want is something like a charcoal stick with undo". Or you have people who want things more like an engineering drawing. I

occasionally do text-based art with GIMP. I'm more likely to use Inkscape because it has slightly better typography. And these days I'm more likely to use a web browser and CSS, because I can get the OpenType features which I can't in GIMP or Inkscape. If you're a typographer, being able to use the font properly is important to you. GIMP is very, very limited at this time, it really is. But it has it, and it's the only thing you can go back and edit afterwards. So from that point of view… [laughing] I think the biggest two problems I hear with people using GIMP, one of them is they haven't found Single Window Mode. If you're not someone who grew up with the X Window System and applications having 25 windows, you might find it really cluttered and difficult to manage. So then you get switched to Single Window Mode, which I use. Even though I used the X Window System as early as 1988 I think. Simon: It's the default now, at least for 2.9. Liam: I'm glad it's the default. It's not perfect, it was never finished, but it's more approachable. The other default that people often have to change is the tile cache size. If you're working with print images especially, then you need to change the tile cache size to be three-quarters of your physical memory or maybe more, depending on what else you've got running. You want as much in memory as possible without crashing your system, and it can be a hard trade-off. Americo: Something I often discover is that people try to "get" GIMP by trial and error, and I'm not sure if this is the right approach to software as complex as GIMP. I've tried this with Paint Shop Pro and also Photoshop, and I didn't get far either. I always have to turn to the manual. Liam: I believe it's deliberate that you can't in Photoshop. Paint Shop Pro I don't know about, it's been years since I used it. But I suspected – don't know this for sure – I suspect that in Photoshop, what they're trying to do – you use to get this from the Debian community as well – the idea is to make something hard enough, it's a trial by fire. You make something hard enough, that people have to put real emotional effort into learning it. And then they'll stick with it through thick and thin. Because they've put so much emotional effort into learning something so hard. Photoshop and Illustrator have an interface where there's hidden buttons in the toolbox. You hold down the mouse pointer over one of the little squares in the toolbox and eventually a hidden secret drawer opens and more tools appear. There's no way you would discover something like that, right, it's not discoverable. The way that you find it is the manual or a tutorial. [Editor's note: In GIMP 2.10, we ended up adding an option for these "hidden buttons", known as tool groups] So for GIMP, probably the best of the books I've seen for learning GIMP is called "The Artist's Guide to GIMP", and I love that guy's tutorials. He says things like, "To start with, press D on the keyboard to get the black and white default colors. Now press X to exchange them. Now you're drawing with white.". And telling you things like that, mean that the tutorial will work regardless of your previous settings. He's teaching you a way of using GIMP where you don't mess yourself up, and if you don't mess yourself up, you have much more confidence to explore. Cover of 'The Artist's Guide to GIMP', by Michael Hammel So I really love his tutorials. He explains, he says how you can do things by clicking, or by using the keystroke or by using the menu. It's really, really well done. So that's what I have suggested to people who want to learn GIMP, is that particular book. But other people prefer video tutorials. I don't have patience for video tutorials – just tell me! But a lot of other people like them. Aryeom: If I want to make a tutorial, is there any advice on the best way? Liam: I've done some tutorials that people didn't like, and some that they really did like. And the tutorials that people really did like are the ones where you're teaching, and at the same time you're not assuming. So you don't assume that people know jargon, you don't assume that people know special names. If you start off in a tutorial saying "The first thing you've got to do is add an alpha channel to your image, and then divide the hypotenuse by the cosine of the vertex", you've lost three-quarters of your audience. If instead you say, "We need our layer to be transparent so that we can see things in the lower layers through it, and to do this, we have to use a menu item called Add Alpha Channel", then what you've done is you've taught someone what alpha channel means and why they want to use it. So instead of just saying "Do this, do this, do this", if you teach people a little while, you're also writing a tutorial that's more robust. Because it might be that in a future version of GIMP, the menu moves. So you can build in notes.

You can do things like say, “If you can’t find this, you can use the slash key to search for things in the new release. This is how you do that”. And then you might make a tutorial that’s robust and that people follow. Because people get really upset if they find a tutorial for GIMP version 1.2 and they try to follow it and it doesn’t work. So then they find a tutorial for some other image editor and they try to follow that and it doesn’t work, obviously. And they get angry and swear at us, and maybe they come to IRC and say your program is full of worms and your mother was a raspberry tart! And we say, “Why are you saying this?” and they say this tutorial doesn’t work! You can’t write something that’s going to be proof against future completely, but you can write something that’s going to be a little bit robust, and that teaches people and keeps the sense of delight and fun. And if you do that, people will enjoy using your tutorial and learn from it, and you’ll have fun. Aryeom: Thank you for the advice! Liam: I’m hoping to do some tutorials this year for scanning images, working on scanning images from books, old books, published, printed books. Jehan: That’s good – for gimp.org? Liam: Yeah, I hope so. One of the things I’m really hoping for is the XSane project makes a version of their GIMP plug-in that works with the new release, with 2.9, to do high bit depth images. Right now it’s restricted to 8 bits per channel. My scanner can do 16 bits per channel, RGB, it’s A4 or Tabloid size. But in fact XSane crashes if I try to do that. Jehan: So you scan with another software? Liam: I scan with XSane, but I have to save it to a file and then I have to open the file in GIMP. And that’s actually a pain for me, because the point at which I choose the file name, the book is still on the scanner, and my file names are usually the page number in the book and the caption as the file name. But if the book is on the scanner then I can’t see what page number it’s on, so I have to remember beforehand to write down the page name, so it’s a pain. So if I’m scanning six images or something at once with the plug-in, it arrives one after the other in GIMP. And a fabulous GIMP feature, not in any of the other software I’ve used, is that I can be scanning one image with a progress bar going along, cleaning up another one, and saving a third one, all at the same time. And that’s three times the throughput that I get with other software. Happy New Year from Moderne Kunst in Meister-Holzschnitten Band XIII (1897), scanned and restored by Liam Quin I’ve been to places where they’re scanning books professionally, and they have three, four, five computers in a row, with multiple scanners. So you start scanning on one, then you move to the next one, start scanning on that, go back to the first one, start cleaning it up, go to the next one, start that one scanning, go this one, start saving the image – because that other proprietary software could only do one of those things at a time. That’s still true today. So by using the plug-in I get much more throughput, so I can have the scanner going – and it can take up to 20 minutes to scan an image, and it can take up to 5 or 10 minutes to save an image. For GIMP to export an image to PNG for example, can take 5 or 10 minutes if it’s a several gigabyte image going to a hard drive. So, it’s really nice to be able to work effectively with multiple threads. I like that. So I should write some tutorials because GIMP really is better than anything else I’ve used. Aryeom: I will translate it to Korean because I’ve seen someone on a Korean website asking how to scan in GIMP like this. Liam: Okay! Maybe they’ve already written a better tutorial than I will, we’ll see. I’ve been doing it for more than ten years. 1999, I think, I started my website, fromoldbooks.org. So it’s going to be coming up to 20 years old before too long. Some of the books are 500 years old. Jehan: And you’ve been using GIMP since the beginning? Liam: I used GIMP early on. I had a period when I was writing a book and the publisher required me to use Microsoft Office, so I was on Windows. For a lot of that time I was using other software. I actually end up using Paint Shop Pro because it had something like, what GIMP calls a corrective or reverse transformation. Jehan: And GIMP did not have it? Liam: GIMP had it, but GIMP wasn’t doing well on Windows at the time. And the other software I had, Photoshop, did not have anything like a corrective mode that was as useful. GIMP’s corrective mode is actually better because it’s a grid and not just a line. The Photoshop one, as I recall, you draw a line on something that should be horizontal or vertical. But in most scanned images I’ve got, they’re hand-made engravings, there isn’t going to be a definite horizontal or vertical, because it’s an artist’s sketch. So I have

to choose what looks best. I start out by getting the grid roughly right, clicking rotate and seeing what happens. Flatten the image to get rid of the corner artifacts and seeing what it looks like. And if it looks okay – with experience you can most of the time do it first time. One of my few patches to GIMP in fact, was to change the undo history to say what the angle was. This way, I can do an undo, even half an hour later, I can see what the angle was and I can say, “Well, that was just a bit too much, I’ll try reducing it just a bit and try again”, and usually second go I get it right. And that’s an example of it being an open source free/libre software. I was able to contribute a patch, which Mitch kindly rejected, and I was able to redo it, and get it in the right format. And he incorporated it – and then I think he wrote it even more. Looking at the code yesterday I discovered – it’s a year or two since I did the patch – I looked at the patch and discovered it’d been rewritten even more, which is good. But that simple patch has saved me, cumulatively, hours and hours of work, just knowing that information. Because I might do two or three rotations in a sequence, then go back and undo them and do a single rotation, because you degrade the image slightly when you rotate it. And if it takes 10, 15 minutes to do a rotate of a large image, then saving two or three attempts at rotation everyday has saved me a lot of time, you know. Screenshot of Undo History with Rotation Angles shown, from Liam’s patch So that’s all part of why GIMP is so fabulous. That, and being able to come onto the IRC channel and say a particular thing is really slow. The first time I did that, it was Sven at the time years ago, he couldn’t believe it was taking me 20, 25 minutes to rotate an image. He looked at the code, and he had me do some profiling, and then he said “Oh”, and made a one line fix. And it went down from 20 minutes to 2 minutes the next time I recompiled GIMP, ten minutes later. Same day! A fix on the same day. In the libre graphics world, in the free software world, that’s not unusual. In the “I’ll make a support ticket with my vendor” world, it’s not unheard of but it’s pretty rare. So that’s been a real plus. Sometimes the developers will say “No, we can’t make that faster because…”, or “We won’t, because we’re going to replace it”. But there’s been a few occasions when GIMP has been faster because I’ve gone into the IRC channel and said I profiled this, or here’s a patch, or can someone fix that. So it’s been fabulous. Jehan: I think we’ve asked most questions. Is there anything you’d like to say that we didn’t ask you? Liam: Yeah. The biggest thing I think we have to do, is get a message out to the world that GIMP is perfectly suitable for professional use. It’s aimed at professional use, and people are using it professionally, successfully. It’s not a second choice, it’s a first choice. It’s not because I can’t afford this other program, or because my principles say I don’t eat meat, or I don’t use programs beginning with P. It’s a first choice – it’s actually better for what I’m doing then any other program I’ve used, and I’ve used a lot. So we need to get that message out a lot more, and we need to have more confidence in talking about GIMP. We’re close to it, we see all the problems, we have visions that we know are not happening, and yet, we forget sometimes that we’ve got something really really good there. And we need not be ashamed to say that. Aryeom: Why do you think people are sometimes ashamed or don’t have confidence to use GIMP? Liam: I knew a chemist once who worked in a jam factory, where they made marmalade and jam. And he saw what the jam did to the steel containers where they mixed it. And they had huge steel vats like the size of a small building, like a big mixing bowl. And the jam is highly acidic, and it would eat the steel. And he said he would never eat jam again, after seeing how jam was made. But you know, if you make jam at home it’s no better. It’s no better than any other food, or worse – alright, it may have too much sugar in it. Or it’s like the sausage maker who knows what goes in the sausage. We’re aware of all the problems, and we’re aware that we have visions of how GIMP could be in some other universe, and it isn’t. But that’s okay, it’s what it is. The fact we can say, the text tool could be improved so that after you scale the image, text is still text for example. We can look at that and say “Yeah, that’d be fairly easy to fix but we’re busy”. But we still have a better text tool than a lot of other programs, you know, even without making any other changes. We’re too busy to go look at other programs – and perhaps worried too about intellectual property, about using someone else’s design. But the truth is, GIMP is better than, in many ways – not perfect, not saying that – but it’s better in many ways than

we realize. Links Liam’s company, Delightful Computing Liam’s other company, From Old Books

- Hylke Bons: Icon for Ratio (2026/09/30 00:00)
  Week 31 This week's icon is for Mark Wyner's project: Ratio: "Validate color contrasts" Check out all weekly app icons created so far in the gallery and follow my icon creation adventures as they happen (including sketches) on the Fediverse. Need icons? I love designing icons and am happy to contribute them free of charge when your project is Free and Open Source. Join as a community sponsor to maintain a steady supply of app icons to the Linux ecosystem (every little helps!).
- Code of Conduct Committee: Announcement: Missed CoC reports (2026/09/29 14:26)
  After connecting some dots, it came to our notice that some conduct reports written through the conduct.gnome.org form might have been silently lost. If you have sent a report between Jan 5th 2026 and Sep 28th 2026 and got no response for it, we invite you to send it again, either through the form or conduct@gnome.org. How In Jan 5th, GNOME sysadmins introduced a filter to the CoC report form to prevent attempts of SQL injection. The filter was however overeager, filtering out messages containing certain natural English words that matched a subset of the SQL syntax. We do not know the full impact of this filter yet. After the issue came to our notice the GNOME sysadmins quickly fixed it on Sept 28th. We appreciate GNOME sysadmins’ diligence and the speedy fix. Contact us If you’ve sent a message in this time period through the conduct.gnome.org form and are missing a response from the CoCC, we invite you to send the report again. It sounds likely that it was filtered down and missed by the CoCC. The CoC Committee
- Hans de Goede: Snapdragon X1 laptop kernel COPR for Fedora 45 (2026/09/28 13:28)
  I'm happy to announce the availability of a COPR repository which rebuilds Fedora 45 kernels with the patches from the qcom-laptops git branch added. This is a branch where various upstream pending patches with Snapdragon laptop improvements are gathered. For the exact contents of the latest kernel from this COPR see this Fedora kernel git-repo fork.Installing the kernel from this COPR adds camera support for various X1 laptop models, adds bluetooth support for the ThinkPad T14s as well various other improvements. Most of these are expected to land in the 7.4 kernel. comments
- Hans de Goede: Snapdragon X1 laptops improvements in Fedora 45 (2026/09/28 13:06)
  After the initial work to make Fedora live iso media boot OOTB on Snapdragon X1 laptops in Fedora 44, I've continued working on improving the Fedora experience on these laptops for Fedora 45. For Fedora 44 a long list of workarounds was necessary, see the Snapdragon laptop install instructions on the wiki.For Fedora 45 various improvements have been made in the mainline kernel and I've been working on improvements at the initramfs generator / distro level. Together these result in a much smoother experience, see the install instructions for Fedora 45 on the wiki. A few workarounds are unfortunately still necessary for Fedora 45. For Fedora 46 I hope that the Fedora aarch64 live iso media will just work on X1 laptops.Note that this blog post is about Snapdragon X1 (Elite,Plus) laptops. Support for Snapdragon X2 laptops at the same level as the current X1 laptop support is quickly coming together in the mainline kernel. But the 7.2 kernel used for the Fedora 45 beta and release isos is still missing some bits and Devicetrees, so Fedora 45 will not work OOTB on Snapdragon X2 laptops. comments
- Arun Raghavan: On LLMs and free software (2026/09/27 21:25)
  Like many, I’ve been thinking a lot about the impact of LLMs on the free software community. In the GNOME community, there have been a couple of posts on Planet GNOME and Discourse, and assorted heated discussions on Matrix and Mastodon. We are not unique in this – there are similar conversations in the KDE and Debian communities as well. I think we should be actively working on figuring out how best we can adapt to

the new world we find ourselves in, and use LLMs for the good of GNOME. The argument being made Some folks in the community would like to have the GNOME project refuse to accept LLM-generated content (be it code, bug reports, or other forms of contributions). The concerns are not unfounded. LLM-generated content can often be verbose, annoying to read, and downright incorrect. This is very dependent on specific models and how they're used, and the state of the art is constantly changing. There are other nuanced arguments around the subject, but I'm skipping them in the interest of brevity. The thrust of the argument is the immediate increase in maintainer load, which is already a matter of concern in the project. I think this is valid, but I think we have to search for solutions to address this problem (for example, the Linux kernel project has Sashiko). Inevitability In my own experience, I have used LLMs to write new tools that I would not have had the time for. I have used them to learn about topics and codebases that would otherwise have taken me much longer to navigate. Using these tools, I have been able to solve some pretty non-trivial problems. In the process, I am also learning the limits of the LLMs, what kind of usage makes sense in what kind of context, and how not to lose the process of critical reasoning while working with them. In every conversation I have had with people across various parts of the software industry, the experiences are similar, and the process of software development is changing. That means that we also have to change the ways in which we interact with each other in building the software that we care so deeply about. We do not need to eject our values to do that. Access Like me, other people are able to use LLMs to build prototypes, write patches, and as a learning tool. This is especially useful for areas where we don't have good documentation. Sometimes the patches or the learning are wrong, but that is not terribly different from reading the code and learning as one often has to do. And as with any nascent tooling, we are still building the right mental models to use for the process. This makes contributing to GNOME more accessible. The arcana of software development are suddenly not in the way of getting something done. Perhaps we could embrace the loss of those barriers and figure out how to include more contributors without adding additional burden to our maintainers. Accessibility There is so much we are yet to realise with LLMs. We could have LLMs (on-device, or otherwise) do: Live captions as hearing assistance Translations for non-native speakers Image descriptions as visual assistance Some of this is table stakes now on other platforms. We have a history of continuous improvement to the desktop accessibility stack, and LLMs unlock a lot of new possibilities to raise the bar on what we are able to provide our users. Keeping GNOME GNOME There are a lot of details missing here – what would the processes look like, how do we keep the infrastructure free (open models? with open datasets?), and so on. But for that, we need to agree on a direction. As a community, we have always been deeply invested in the human aspects of the software that we build. Even if the state of LLMs is frozen at the current level, we are seeing a sea-change in the process of building software, and we cannot hide from it. As with other changes before this (the inception of the project, the adoption of the HIG, GNOME 3), it behooves us to play a positive role in defining how we build software for humans in the future.

- Morten Welinder: LLMs: Vigilatism is not the Answer (2026/09/27 19:23)
  Jordan Petridis evidently has strong opinions on the use of LLMs. He's welcome to those. What's not welcome is when it escalates into vigilantism and defacement of bug reports. Specifically for me, this bug where a tag "Probabilistically Automated" was added. That tag has a meaning of Usually accompanied by a lack of proper testing, and finalizing patches based on theoretical intended behavior rather than correctness of code. I am reading that as screaming "It's the work of the Devil!" in a shrill voice. The naming of the tag likewise suggests religious fanaticism. The LLM work in the bug report was (1) specifically requested, (2) excellent quality, and (3) very helpful given that the underlying problem does not happen on my machine. If you can't be bothered to actually assess the quality instead of hiding behind "usually" then you are not adding anything positive and should stay away. When asked for an explanation, none was provided. A comment on his blog post was, as far as I can tell,

moderated away. (I do agree that poor-quality LLM-generated bug reports exist. Do they ever. A tag like the above is not part of the solution to that, whatever it may be.)

- Jordan Petridis: Introducing Toolpak (2026/09/26 11:37)
  Most modern operating systems (including iOS, Android, macOS, and ChromeOS) have been using image-based architectures for a while, and in recent years we've been making progress towards this on the desktop side as well. It's a necessity if we want to provide the security, usability and reliability that people expect from their devices nowadays. However, broadly speaking the adoption of image-based designs has been limited in the free desktop world so far because there are still a few major gaps. While Flatpak and Flathub have more or less solved the app distribution on these systems, the developer-facing story still looks much more incomplete. Traditionally, people have been developing against the host OS they are running. Assuming you're working on something like NetworkManager and need a new dependency, you'd just "apt install" it globally on your system, and then configure/compile against that. Similarly, if you need a command-line utility, compiler toolchain, or other build dependency, you'd also install them from the distribution repository. In an image-based world this approach doesn't work, and people have come up with various ways to address this. RPM-Ostree and Package Overlays Fedora Silverblue can be extended similar to package-based distros with rpm-ostree overlays, but the issue with this approach is that overlayed packages can completely break the system in unexpected ways or stop it from further updating. This is why the next iteration of Silverblue is based on "bootc", and will be explicitly avoiding this paradigm. Same issue are present with other package overlay approaches as well. Monolithic "Development" Overlay The approach used by Android, iOS, Windows, et al is to have a single monolithic overlay with "system development tools". Developers can install this overlay, and it provides all the utilities people developing the system itself need. GNOME OS does something similar: There is a "developer" system extension that overlays the toolchain used for building the OS on top of the user-facing OS image. However, this is a finite list of utilities needed specifically to build the OS. As such, it can not cover the long tail of development and debugging tools developers in different areas need (e.g. kernel development). Toolbox Another approach is trying to replicate the same traditional package-based experience inside a container. Toolbox and distrobox are examples of this. Unfortunately, with this approach you are still relying on the same old package infrastructure, while also being inside a more restrictive container environment that none of the tools expect to be run in. As such, you often run into limitations when developing system components and need to bypass the container layers in order to debug the system itself. Homebrew Homebrew provides an independent tool chain to develop against, but in order for homebrew binaries to be usable directly in the terminal, they have to be prioritized over system binaries. This means that the system can break if there is a mismatch between what the system expects and what homebrew provides. For example, you install QEMU, but it overrides the GLib your system uses for everything else. There are other architectural issues with homebrew, but in my opinion this alone disqualifies it for system development. Flatpak Lastly, we have Flatpak. Like Toolbox it uses "containers", so the same issues and limitations are also present here, but it's even more restrictive because the assumption is that apps will use portals to access system resources such as directories and devices. Flatpak was designed specifically with desktop apps in mind, and its architecture and integration points are not well suited for command line apps. Somewhat tangentially, there are apps that can't be made to fully work with Flatpak as it is today, particularly debugging utilities and IDEs. While it's technically possible (GNOME Builder and the Ptyxis terminal are proof), older applications were not designed with sandboxed resources in mind, or have not adapted to this model yet. What Could We Do Instead? While we would love for everything to be sandboxed and confined, that is sadly not yet possible. All the existing approaches that attempted to containerize tooling, run into the same conflicts between the desired functionality and the restrictions inherited by this approach. This is a topic we will discuss in more

depth later on. Additionally I believe there are two different use cases here. The build tooling/toolchain used by projects, and the developer utilities. Project Build Setups One aspect of this is the build setup used by individual project, which would ideally be something more deterministic, and containerized like Buildstream, flatpak-builder, bazel or even nix, instead of the old status quo “apt install -yqq gcc meson libone-devel libtwo-devel”. This is also a topic for another day though. Developer Utilities The other aspect, and what I want to focus on today, is that a lot of developer utilities that people rely on can’t realistically be run in a confined environment (such as a container) without a near complete rewrite. We need a way to make things like strace, ripgrep, and qemu available. Shipping them in the host system is one option, but there is a real long tail issue here. You can’t (and don’t want to) provide every single utility any developer might need. Thus we need them to be shipped independently, and as such, not attached to the host OS. Toolpak If we were to take a fresh look at this issue and design something from scratch, what would we want it to look like? Over the past months we’ve had a number of discussions on these topics, and it feels like we’re finally converging on a solution would address most people’s concerns and needs. These are the important properties we would want: Tools should be independent from the host OS, and they should not break the host OS if something goes wrong. They need to be able to access every resource in the system, much like today. A large catalog of existing tooling, like we find today in distribution repositories today. Tools work out of the box, and will be fully functional, as it’s not feasible to modify all of them. Things like shelling out to other tools should work like it does now. Here is what I imagine an implementation, which we will call Toolpak, would look like: Discoverable Disk Images (UAPI.3) as the image format. This will provide us state of the art security practices, like Verity. It will also give us a good base for allowing Reproducible builds, given that the build tooling permits. A mount namespace for the binaries. The /usr and /app split from Flatpak is a great idea and we should also steal it much like portable services now can use Base/Runtime images! /usr should be provided by the “Runtime Image” and shared among tools. And /app will be the main contents of the Tool image. Tools have unrestricted access to everything else. This should also satisfy the “no need to port” requirement (Some edge cases where the mount namespace will conflict, but they are minor). We should prepend the tools, to the PATH of the user (ie. our binaries will override the system ones), but it should avoid breaking the system as you can only override the system binaries. Said binary will then setup the mount namespace and execute the real binary from inside the image. This avoids messing with the linker and shared libraries. Unless you overriding the system GNU Tar with the BSD Tar (or other incompatible cases of the same binary), things should work fine. More research is needed to explore what other safeguards will be required. Tools can not depend on other tools. One of the issues with traditional distributions is dependency resolution and package management (this is subcategory of a heavily discussed topic, I also talked about it in my LAS talk from 2025. We should avoid making individual packages or having dependencies between tools, to avoid all the unnecessary complexity that comes with it. Tools will bundle all their dependencies. This will ensure that they will always execute against the environment they were tested against. It additionally allows tools to bring their own versions of libraries that might be present in the system, without conflicts. We mentioned the /usr and /app split above. Great user experience to discover and install tools. There should be a flagship “app store” and ideally the tools should be packaged and distributed by the developers themselves, similar to Flathub. No more “download this static binary and chmod +x it”. Tools have complete and arbitrary access to the system, so it’s crucial that it not be simple to install random images. They will be have to be signed with a trusted key, and verity checked at runtime, and be thoroughly reviewed before appearing on the flagship “app store”, and all the other practices we should expect from distributing software securely. Apps that can be packaged using Flatpak should be rejected. Unless they are less functional, like IDEs on Flathub. In order to be adopted, it will also have to come with tooling that will make it easy to build said Toolpaks. Here are some specific properties the build tooling should have: It should be easy to orchestrate builds of your tool and its dependencies. Great

caching for all build artifacts in a Content Addressable Storage. Reproducible by default, with tooling to easily verify the output. Can be used for both local development and composing the final image. Handles all the licensing/SBOM/etc requirements. A Buildstream plugin or wrapper around it should satisfy all these needs. Next Steps While this topic has been discussed for years, there’s now concrete work towards a prototype as part of a Prototypefund project. We plan to share more on this in the coming weeks and are interested in feedback from the wider community. In the mean time, if you have any other feedback, find us in #gnome-os:gnome.org on Matrix, or leave a comment.

- Diego Escalante Urrelo: LLM Policies: Progress At All Costs (2026/09/25 15:00)
  GNOME and KDE have started to consider LLM policies, and we should talk about what this is really about. KDE caught everyone's attention first by igniting a flame war with an LLM-friendly draft that ended up being deleted, causing a few bans, and having a bunch of people go full "Some of you may die, but that is a sacrifice I am willing to make". GNOME has not proposed anything official, but some teams (gnome-calendar, loupe, libadwaita, gnome-software, Circle, among others) already have strong policies in place, and there is now an informal draft to ban all LLM contributions to GNOME projects and infrastructure. However I believe these discussions are not about nitpicking workflows but rather about the raison d'être, the reason to be, of FLOSS projects. Communities Or Completionism My take is that there are currently two ways of thinking about why FLOSS exists, or should exist. So far, both sides have coexisted but the growing acceptance of LLMs into some developer workflows has disrupted the balance. We can call one of these sides "collectivism". This frames FLOSS to be about accomplishing things together, enjoying the journey and bonds that big goals tend to create. The fun is the collective effort and challenge. Overcoming language and social barriers is part of the reward. "The journey is the destination", "FLOSS is the friends we made along the way", etc. On the other side there is "completionism", where FLOSS is "just a product" and its only goal is to always be better, faster, safer. Any fun to be had is in individually solving technical problems and requirements. Social bonds may happen, but colleagues are more coworkers than community. This is a "100% allglitches alltricks" TAS speedrun. The "we are apolitical", "we only care about the code" view. My assessment is that the collectivist framing finds FLOSS primarily a social exercise that rewards you with experiences, bonds, and ideas outside of your niche interests. Some times you even get good software as a bonus! The second framing sees FLOSS as a tool to scale the complexity of your individual computer interest, like graphics or security. FLOSS is a convenience compared to manually rebasing patches and forks all the time. The problem we are facing is that LLMs have given the second group a lever to stop giving the collectivist framing any room. When the LLM can get you 80% of the way to your goal, there is no need to "waste" time in mentoring, discussion, or convincing others. The temptation compounds if you are considered an expert in your field. You can surely fill in the last 20%, right? Is anyone going to challenge not only the machine, but also the expert? Progress At All Costs Since LLMs present themselves as neutral, and dispassionate, opposition to their output becomes opposition to objective progress: bug fixes, security hypotheticals, features. Progress is whatever the LLM, under my own careful eyes, says it is. Interactions with others become formalities, since the LLM is simply boosting my own, already expert and close to infallible, output. Right? Unfortunately this "expert slop", where expert is a self-perceived title, carries a corporate framing that damages interactions. Others become, at best, fungible coworkers, and, at worst, annoying speed bumps in the race to 100% completion of any software interest the expert has. No more mentoring, debating, flame wars. "Progress" is the only goal. The line must go up. There is more to say about how this machiavellian framing causes far more important harms and externalities, in the name of LLMs themselves, or apparent LLM-assisted progress. Think of any group and you will find out they have been handed part of the bill for these externalities: The people who will lose their homes, schools, lives, in the Global South because of El Niño 2027 The towns destroyed by data center illegal and irresponsible practices The price of all consumer goods going through the roof because of financial speculation The pensioners

who might lose their pension when the bubble finally pops. These are the real costs in the "Progress At All Costs" that LLMs bring into FLOSS. These are the people who will pay the bill, behind the scenes and far from our screens, so that some big brain engineers can avoid reading documentation, writing boilerplate, learning unfamiliar code, or, worse, working with others. FLOSS As Principled Software Almost ten years ago Allan Day described GNOME as Principled Software because of its commitment to always doing the right thing, in code or design, because it was the right thing and not because of ease, pressure, or hype. I believe this is why so many other FLOSS projects have always looked at GNOME for guidance on what good FLOSS should be. This discussion is just another opportunity to continue to meet this expectation. Recent discussions have shared similar sentiments like reminding us that we do book clubs because we want to read and enjoy books, not to just discuss over summaries because it is "more productive". That when pressed to accelerate FLOSS, to make the line go up, we have to ask for whom do we want to be more productive, efficient, faster?. And that every decision is political and affect other people around you. We already know that LLM productivity is not real, just a self perception, that LLMs are just a fairy tale to maintain tech stocks hypergrowth, by farming engineers for engagement, and the latest in a series of attacks to commoditize tech workers. Knowing all this, are we still going to play along with big tech's lies and exploitation? Or, are we going to make another principled stand? GNOME did not need LLMs to produce 30 years of creative engineering, design, localization, inclusion, and collaboration that has been shared with people around the world. It does not need to throw away this incredible legacy simply because LLMs happen to farm our worst individualist impulses. We came this far without compromising our principles, let's not start now.

- Juan Pablo Ugarte: Casilda 1.6 Released! (2026/09/24 16:25)
  DnD Release I am very happy to announce a new version of Casilda! A simple Wayland compositor widget for Gtk 4 originally created for Cambalache This new version brings Drag&Drop and clipboard support / integration with the host compositor which means you can drag something from a host application and drop it on an application window embedded in a Casilda compositor widget and vice versa! After a very rugged GUADEC presentation where I tried to show how to create simple Gtk application from slides made with Cambalache using a Casilda compositor to embed GNOME Builder and Cambalache itself I decided to fix all the issues I found so I could redo the presentation and upload it as a video. This is a screenshot of the slider window, inside it there is a Casilda compositor (green line) used to embed a Cambalache instance which uses another CasildaCompositor (red line) to preview the UI. Things that possi-bly could go wrong and did: Clipboard (Could not copy paste snippets) Keyboard numeric pad Keyboard modifiers in pointer events (I could not use Alt+Click to force create widgets) Example application connected to host compositor instead of Casilda widget Gnome Builder fullscreen window state broken Popover tooltips crash After fixing all the bugs and keyboard support I started working on adding Clipboard integration with the host. Clipboard support Integrating the clipboard between Casilda and Gtk was fairly easy since all I needs to be done is copy all the clipboard data provided by wlroots wlr_seat::request_set_selection event to GdkClipboard and call wlr_seat_set_selection() on GdkClipboard changed signal. Wlroots client to Gtk In other to do this I created a new GdkContentProvider that uses a wlr_data_source as the source of the data and set the provider as the content of the clipboard with gdk_clipboard_set_content(). Internally when a clients wants to get data from the clipboard the new provider creates a unix pipe to read data from wlroots by writing to the pipe with wlr_data_source_send() and reading form the other end using g_output_stream_splice_async(). Gtk to wlroots to client To go in the other direction I created a new wlr_data_source that uses a GdkClipboard as source and set selection with wlr_seat_set_selection() Internally gdk_clipboard_read_async() is called to initiate the data read when wlr_data_source send() method is invoked to send the data to the client. Drag & Drop support Integrating D&D was not as easy as the clipboard. Setting it up to work within Casilda is easy

enough other than wiring up some events all I had to do was draw the drag icon surface. The first thing I did after getting the drag surface was draw it at the pointer coordinates which produced blurry results since pointer coordinates are fractional which means they do not always align with the pixel grid. That was all nice and good but now I needed to start integrating the drag with Gtk (host compositor) so right after calling wlr_seat_start_pointer_drag() I created a GdkDrag with gdk_drag_begin() and used gtk_drag_icon_set_from_paintable() to set the drag icon and I noticed it was still blurry!!! After a minute of confusion I realized that Gnome Shell (mutter) must be making the same mistake I did before so I decided to see if I could fix it! Thanks to Michel Dänzer for guiding me and reviewing my MR, Gnome Shell 51 should have sharp drag icons! Now back to integrating DnD with the host compositor. These are all the different use cases that are needed to make D&D work transparently across host and guest compositors: Casilda client to Casilda client Casilda client to Host client Host client to Casilda client Of course wlroots only contemplates the first use case, the other two cases are specific to Casilda. Casilda to Host So far we can detect when a client start a drag, let wlroot handle it and create a GdkDrag to let Gtk initiate another drag at the host level. This means that when a client embedded in a CasildaCompositor initiates a drag there are two simultaneous drags at the same time, in the guest and host compositors. Keep in mind Casilda always delegates drag icon rendering to the host compositor. During normal operation Casilda gets events from an event controller, but while on a drag operation events are not sent to the client at least not in the normal way, for that you can use a GtkDropTargetAsync and connect to the various signals for example I use drag-motion to forward events to the client. Host to Casilda When a Drag is initiated in the host compositor Casilda reuses the GtkDropTargetAsync created to track motion events and connects to drag-enter signal to create a proxy drag source in wlroots and synthesize a button release event on GtkDropTargetAsync::drop to trigger the drop on the guest side. Here is a screencast off all the different use cases in action. As you can imagine getting all this to work together correctly is not trivial and I expect to be subtle bugs in different corner cases so please if you find one of those file a bug and include a screencast if possible. Release Notes Add DnD and clipboard support Add support xdg_foreign protocol Fix modifiers in pointer button press events Add keyboard Caps/Num/Scroll Lock support Fix popup of popup crash Fix maximized/fullscreen state handling Fix modifiers flags creation (Evgenii Danilin) Drop cursor surface listeners when the surface is destroyed (Evgenii Danilin) Close dup’d plane FDs when a dmabuf import fails (Evgenii Danilin) Advertise a valid output scale to avoid a scale-0 assert (Evgenii Danilin) Unref the wayland GSource (Evgenii Danilin) Meson config cleanup (Val Packett) Use gtk api for snapping to device pixel grid Fixed Issues #20 “SIGSEGV in server_request_acvtivate” Where to get it? Source code lives on GNOME gitlab here git clone https://gitlab.gnome.org/jpu/casilda.git Matrix channel Have any question? come chat with us at #cambalache:gnome.org Mastodon Follow me in Mastodon @xjuan to get news related to Casilda and Cambalache development. Happy coding!

- Jordan Petridis: The GNOME LLM Policy That I Want (2026/09/23 06:02)
  KDE is on the news because of a controversial proposal to define an official “AI” (LLM) policy (archived link). Other projects have tried their hand at similar policies and stances but, in my opinion, they miss the mark about the goal of such initiatives. I think that the point of these statements is shaping social norms and not micro-managing developer workflows. They should be about signalling what kind of behavior we want, and what kind we reject. This proposal does not go into detail about the many problems that LLM have caused to society, workers, the environment. It goes without saying that all these ills are fundamentally opposed to the humanist spirit of GNOME. With all that in mind, here is what I personally think a GNOME LLM policy could be: A GNOME Project LLM Policy The GNOME Project prioritizes the social and human aspects of collective software creation. Therefore: 1. LLMs ("AI") can not be used to create or modify anything submitted to GNOME, or hosted on GNOME infrastructure. You might be asked to prove your code meets this requirement. You might be banned for trying to circumvent this policy. Example Guidelines for

Contributors These are just a draft of the kind of criteria one could use to evaluate if a submission fits the policy. You must be able to personally reason and explain your changes You must be able to demonstrate knowledge of the problem space you are working on You must solve the underlying issue, not just its symptoms You must respect the time of fellow contributors You must not impersonate yourself through chatbots, agents, or other automated systems This Is About The Future Of GNOME GNOME is not just software that happens to ship every six months. That is just a delusion we have been holding up for the last 30 years to keep our loose group of colleagues, friends, and acquaintances, together. GNOME exists as a collective that find joy in reaching beyond our individual limitations to achieve something bigger. These people, this joy, are the whole point of the project. Contributors are not payroll, a liability, that we hope to downsize next quarter. “Come do free labor for a handful of corporations by reviewing chatbot output in your free time” is not an attractive proposition to young talented people in 2026. If we want GNOME to continue we need to create an attractive and inviting social space where people are valued as people. Just like the Foundation is moving to individual donations to stop depending on just a handful of companies, we need to look for the next 100 people that will donate a tiny bit of their time, instead of hoping that corporations will keep 10 overworked engineers on staff. We already have seen how companies will happily abandon a whole chunk of GNOME on a whim. GNOME is not just software, and it should protect the social and human aspects that make it special. Our success metric is the community and social bonds we create. In the most literal sense GNOME is about the journey and the friends we make along the way. Free Palestine. FAQ How do you enforce this? You can not. People will still send LLM generated code. This policy makes it explicit that we do not welcome these careless submissions. We have a Code of Conduct that is 80% about telling other people what our values are, and 20% about handling unwanted behavior (“enforcing”). This is similar. What if people simply lie about not using LLMs? This is the same problem as authorship, in the copyright sense. Whenever we receive new code we have to assume that “beyond a reasonable doubt” said code has been authored by the person contributing it. We make our best guess. The attached guidelines are a suggestion to make these new guesses. Ok. But what if people are really good at lying? This policy is about the majority that will not even try to lie. See previous questions.

From:
https://wiki.tromjaro.alexio.tf/ - **TROMjaro wiki**

Permanent link:
https://wiki.tromjaro.alexio.tf/doku.php?id=news:planet:gnome

Last update: **2021/10/30 11:41**