Gnome Planet - Latest News

- Matthew Garrett: SSH certificates and git signing (2026/03/21 19:38)
  When you’re looking at source code it can be helpful to have some evidence indicating who wrote it. Author tags give a surface level indication, but it turns out you can just lie and if someone isn’t paying attention when merging stuff there’s certainly a risk that a commit could be merged with an author field that doesn’t represent reality. Account compromise can make this even worse - a PR being opened by a compromised user is going to be hard to distinguish from the authentic user. In a world where supply chain security is an increasing concern, it’s easy to understand why people would want more evidence that code was actually written by the person it’s attributed to. git has support for cryptographically signing commits and tags. Because git is about choice even if Linux isn’t, you can do this signing with OpenPGP keys, X.509 certificates, or SSH keys. You’re probably going to be unsurprised about my feelings around OpenPGP and the web of trust, and X.509 certificates are an absolute nightmare. That leaves SSH keys, but bare cryptographic keys aren’t terribly helpful in isolation - you need some way to make a determination about which keys you trust. If you’re using someting like GitHub you can extract that information from the set of keys associated with a user account1, but that means that a compromised GitHub account is now also a way to alter the set of trusted keys and also when was the last time you audited your keys and how certain are you that every trusted key there is still 100% under your control? Surely there’s a better way. SSH Certificates And, thankfully, there is. OpenSSH supports certificates, an SSH public key that’s been signed by some trusted party and so now you can assert that it’s trustworthy in some form. SSH Certificates also contain metadata in the form of Principals, a list of identities that the trusted party included in the certificate. These might simply be usernames, but they might also provide information about group membership. There’s also, unsurprisingly, native support in SSH for forwarding them (using the agent forwarding protocol), so you can keep your keys on your local system, ssh into your actual dev system, and have access to them without any additional complexity. And, wonderfully, you can use them in git! Let’s find out how. Local config There’s two main parameters you need to set. First, 1 git config set gpg.format ssh because unfortunately for historical reasons all the git signing config is under the gpg namespace even if you’re not using OpenPGP. Yes, this makes me sad. But you’re also going to need something else. Either user.signingkey needs to be set to the path of your certificate, or you need to set gpg.ssh.defaultKeyCommand to a command that will talk to an SSH agent and find the certificate for you (this can be helpful if it’s stored on a smartcard or something rather than on disk). Thankfully for you, I’ve written one. It will talk to an SSH agent (either whatever’s pointed at by the SSH_AUTH_SOCK environment variable or with the -agent argument), find a certificate signed with the key provided with the -ca argument, and then pass that back to git. Now you can simply pass -S to git commit and various other commands, and you’ll have a signature. Validating signatures This is a bit more annoying. Using native git tooling ends up calling out to ssh-keygen2, which validates signatures against a file in a format that looks somewhat like authorized-keys. This lets you add something like: 1 * cert-authority ssh-rsa AAAA… which will match all principals (the wildcard) and succeed if the signature is made with a certificate that’s signed by the key following cert-authority. I recommend you don’t read the code that does this in git because I made that mistake myself, but it does work. Unfortunately it doesn’t provide a lot of granularity around things like “Does the certificate need to be valid at this specific time” and “Should the user only be able to modify specific files” and that kind of thing, but also if you’re using GitHub or GitLab you wouldn’t need to do this at all because they’ll just do this magically and put a “verified” tag against anything with a valid signature, right? Haha. No. Unfortunately while both GitHub and GitLab support using SSH certificates for authentication (so a user can’t push to a repo unless they have a certificate signed by the configured CA), there’s currently no

way to say “Trust all commits with an SSH certificate signed by this CA”. I am unclear on why. So, I wrote my own. It takes a range of commits, and verifies that each one is signed with either a certificate signed by the key in CA_PUB_KEY or (optionally) an OpenPGP key provided in ALLOWED_PGP_KEYS. Why OpenPGP? Because even if you sign all of your own commits with an SSH certificate, anyone using the API or web interface will end up with their commits signed by an OpenPGP key, and if you want to have those commits validate you’ll need to handle that. In any case, this should be easy enough to integrate into whatever CI pipeline you have. This is currently very much a proof of concept and I wouldn’t recommend deploying it anywhere, but I am interested in merging support for additional policy around things like expiry dates or group membership. Doing it in hardware Of course, certificates don’t buy you any additional security if an attacker is able to steal your private key material - they can steal the certificate at the same time. This can be avoided on almost all modern hardware by storing the private key in a separate cryptographic coprocessor - a Trusted Platform Module on PCs, or the Secure Enclave on Macs. If you’re on a Mac then Secretive has been around for some time, but things are a little harder on Windows and Linux - there’s various things you can do with PKCS#11 but you’ll hate yourself even more than you’ll hate me for suggesting it in the first place, and there’s ssh-tpm-agent except it’s Linux only and quite tied to Linux. So, obviously, I wrote my own. This makes use of the go-attestation library my team at Google wrote, and is able to generate TPM-backed keys and export them over the SSH agent protocol. It’s also able to proxy requests back to an existing agent, so you can just have it take care of your TPM-backed keys and continue using your existing agent for everything else. In theory it should also work on Windows3 but this is all in preparation for a talk I only found out I was giving about two weeks beforehand, so I haven’t actually had time to test anything other than that it builds. And, delightfully, because the agent protocol doesn’t care about where the keys are actually stored, this still works just fine with forwarding - you can ssh into a remote system and sign something using a private key that’s stored in your local TPM or Secure Enclave. Remote use can be as transparent as local use. Wait, attestation? Ah yes you may be wondering why I’m using go-attestation and why the term “attestation” is in my agent’s name. It’s because when I’m generating the key I’m also generating all the artifacts required to prove that the key was generated on a particular TPM. I haven’t actually implemented the other end of that yet, but if implemented this would allow you to verify that a key was generated in hardware before you issue it with an SSH certificate - and in an age of agentic bots accidentally exfiltrating whatever they find on disk, that gives you a lot more confidence that a commit was signed on hardware you own. Conclusion Using SSH certificates for git commit signing is great - the tooling is a bit rough but otherwise they’re basically better than every other alternative, and also if you already have infrastructure for issuing SSH certificates then you can just reuse it4 and everyone wins. Did you know you can just download people’s SSH pubkeys from github from https://github.com/<username>.keys? Now you do ↩ Yes it is somewhat confusing that the keygen command does things other than generate keys ↩ This is more difficult than it sounds ↩ And if you don’t, by implementing this you now have infrastructure for issuing SSH certificates and can use that for SSH authentication as well. ↩

- Sam Thursfield: Status update, 21st March 2026 (2026/03/21 18:29)
Hello there, If you’re an avid reader of blogs, you’ll know this medium is basically dead now. Everyone switched to making YouTube videos, complete with cuts and costume changes every few seconds because, I guess, our brains work much faster now. The YouTube recommendation algorithm, problematic as it is, does turn up some interesting stuff, such this video entitled “Why Work is Starting to Look Medieval”: It is 15 minutes long, but it does include lots of short snippets and some snipping scissors, so maybe you’ll find it a fun 15 minutes. The key point, I guess, is that before we were wage slaves we used to be craftspeople, more deeply connected to our work and with a sense of purpose. The industrial revolution marked a shift from cottage industry, where craftspeople worked with their own tools in their own house or workshop, to

modern capitalism where the owners of the tools are the 1%, and the rest of us are reduced to selling our labour at whatever is the going rate. Then she posits that, since the invention of the personal computer, influencers and independent content creators have begun to transcend the structures of 20th century capitalism, and are returning to a more traditional relationship with work. Hence, perhaps, why nearly everyone under 18 wants to be a YouTuber. Maybe that's a stretch. This message resonated with me after 20 years in the open source software world, and hopefully you can see the link. Software development is a craft. And the Free Software movement has always been in tacit opposition to capitalism, with its implied message that anyone working on a computer should have some ownership of the software tools we use: let me use it, let me improve it, and let me share it. I've read many many takes on AI-generated code this year, and its really only March. I'm guilty one of these myself: AI Predictions for 2026, in which I made a link between endless immersion in LLM-driven coding and more traditional drug addictions that has now been corroborated by Steve Yegge himself. See his update "The AI Vampire" (which is also something of a critique of capitalism). I've read several takes that the Free Software movement has won now because it is much easier to understand, share and modify programs than ever before. See, for example, this one from Bruce Perens on Linquedin: "The advent of AI and its capability to create software quickly, with human guidance, means that we can probably have almost anything we want as Free Software.". I've also seen takes that, in fact, the capitalism has won. Such as the (fictional) MALUSCorp: "Our proprietary AI robots independently recreate any open source project from scratch. The result? Legally distinct code with corporate-friendly licensing. No attribution. No copyleft. No problems.". One take I haven't seen is what this means for people who love the craft of building software. Software is a craft, and our tools are the operating system and the compiler. Programmers working on open source, where code serves as reference material and can live in the open for decades, will show much more pride in their code than programmers in academia and industry, whose prototypes or products just need to get the job done. The programmer is a craftsperson, just like the seamstress, the luthier and the blacksmith. But unlike clothes, guitars and horseshoes, the stuff we build is intangible. Perhaps as a result, society sees us less like craftspeople and more like weird, unpopular wizards. I've spent a lot of my career building and testing open source operating systems, as you can see from these 30 different blog posts, which include the blockbuster "Some CMake Tips", the satisfying "Tracker Meson", and or largely obsolete "How BuildStream uses OSTree". It's really not that I have some deep-seated desire to rewrite all of the world's Makefiles. My interest in operating systems and build tools has always came from a desire to democratize these here computers. To free us from being locked into fixed ways of working designed by Apple, Google, Microsoft. Open source tools are great, yes, but I'm more interested in whether someone can access the full power of their computer without needing a university education. This is why I've found GNOME interesting over the years: it's accessible to non-wizards, and the code is right there in the open, for anyone to change. That said, I've always wished we GNOME focus more on customizability, and I don't mean adding more preferences. Look, here's me in 2009 discovering Nix for the first time and jumping straight to this: "So Nix could give us beautiful support for testing and hacking on bits of GNOME". So what happened? Plenty has changed, but I feel that hacking on bits of GNOME hasn't become meaningfully easier in the intervening 17 years. And perhaps we can put that largely down to the tech industry's relentless drive to sell us new computers, and our own hunger to do everything faster and better. In the 1980s, an operating system could reasonably get away with running only one program at a time. In the 1990s, you had multitasking but there was still just the one CPU, at least in my PC. I don't think there was any point in the 2000s when I owned a GPU. In the 2010s, my monitor was small enough that I never worried about fractional scaling. And so on. For every one person working to simplify, there are a hundred more paid to innovate. Nobody gets promoted for simplicity. I can see a steadily growing interest in tech from people who aren't necessarily interested in programming. If you're not tired of videos yet, here's a harp player discussing the firmware of a digital guitar pedal (cleverly titled “What pedal makers don’t want you to see”). Here’s another musician discussing STM32 chips and mesh networks under the title “Gadgets For People Who Don’t Trust The Government”. This one does not have costume changes every few seconds. So we’re at an inflection point. The billions pumped into the AI bubble come from a desire by rich men to take back control of computing. It’s a feature, not a bug, that you can’t run ChatGPT on a consumer GPU, and that AI companies need absolutely all of the DRAM. They could spend that money on a programme like Outreachy, supporting people to learn and understand today’s software tools … but you don’t consolidate power through education. (The book Careless People, which I recommend last year, will show you how much tech CEOs crave raw power). In another sense, AI models are a new kind of operating system, exposing the capabilities of a GPU in a radical new interface. The computer now contains a facility that can translate instructions in your native language into any well-known programming language. (Just don’t ask it to generate Whitespace). By now you must know someone non-technical who has nevertheless automated parts of their job away by prompting ChatGPT to generate Excel macros. This is the future we were aiming for, guys! I’m no longer sure if the craft I care about is writing software, or getting computers to do things, or both. And I’m really not sure what this craft is going to look like in 10 or 20 years. What topics will be universally understood, what work will be open to individual craftspeople, and what tools will be available only to states and mega-corporations? Will basic computer tools be universally available and understood, like knives and saucepans in a kitchen? Will they require small scale investment and training, like a microbrewery? Or will the whole world come to depend on a few enourmous facilities in China? And most importantly, will I be able to share my passion for software without feeling like a weird, unpopular wizard any time soon?

- Allan Day: GNOME Foundation Update, 2026-03-20 (2026/03/20 15:42)
  Hello and welcome to another update on what’s been happening at the GNOME Foundation. It’s been two weeks since my last update, and there’s been plenty going on, so let’s dive straight in. GNOME 50! My update wouldn’t be complete without mentioning this week’s GNOME 50 release. It looks like an amazing release with lots of great improvements! Many thanks to everyone who contributed and made it such a success. The Foundation plays a critical role in these releases, whether it’s providing development infrastructure, organising events where planning takes place, or providing development funding. If you are reading this and have the means, please consider signing up as a Friend of GNOME. Even small regular donations make a huge difference. Board Meeting The Board of Directors had its regular monthly meeting on March 9th, and we had a full agenda. Highlights from the meeting included: The Board agreed to sign the Keep Android Open letter, as well as endorsing the United Nations Open Source Principles. We heard reports from a number of committees, including the Executive Committee, Finance Committee, Travel Committee, and Code of Conduct Committee. Committee presentations are a new addition to the Board meeting format, with the goal of pushing more activity out to committees, with the Board providing high-level oversight and coordination. Creation of a new bank account was authorized, which is needed as part of our ongoing finance and accounting development effort. The main discussion topic was Flathub and what the organizational arrangements could be for it in the future. There weren’t any concrete decisions made here, but the Board indicated that it’s open to different options and sees Flathub’s success as the main priority rather than being attached to any particular organisation type or location. The next regular Board meeting will be on April 13th. Travel The Travel Committee met both this week and last week, as it processed the initial batch of GUADEC sponsorship applications. As a result of this work the first set of approvals have been sent out. Documentation has also been provided for those who are applying for visas for their travel. The membership of the current committee is quite new and it is having to figure out processes and decision-making principals as it goes, which is making its work more intensive than might normally be the case. We are starting to write up guidelines for future funding rounds, to help smooth the process. Huge thanks to our committee members Asmit, Anisa, Julian, Maria,

and Nirbeek, for taking on this important work. Conferences Planning and preparation for the 2026 editions of LAS and GUADEC have continued over the past fortnight. The call for papers for both events is a particular focus right now, and there are a couple of important deadlines to be aware of: If you want to speak at LAS 2026, the deadline for proposals is 23 March – that’s in just three days. The GUADEC 2026 call for abstracts has been extended to 27 March, so there is one more week to submit a talk. There are teams behind each of these calls, reviewing and selecting proposals. Many thanks to the volunteers doing this work! We are also excited to have sponsors come forward to support GUADEC. Accounting The Foundation has been undertaking a program of improvements to our accounting and finance systems in recent months. Those were put on hold for the audit fieldwork that took place at the beginning of March, but now that’s done, attention has turned to the remaining work items there. We’ve been migrating to a new payments processing platform since the beginning of the year, and setup work has continued, including configuration to make it integrate correctly with our accounting software, migrating credit cards over from our previous solution, and creating new web forms which are going to be used for reimbursement requests in future. There are a number of significant advantages to the new system, like the accounting integration, which are already helping to reduce workloads, and I’m looking forward to having the final pieces of the new system in place. Another major change that is currently ongoing is that we are moving from a quarterly to a monthly cadence for our accounting. This is the cycle we move on to “complete” the accounts, with all data inputted and reconciled by the end of the cycle. The move to a monthly cycle will mean that we are generating finance reports on a more frequent basis, which will allow the Board to have a closer view on the organisation’s finances. Finally, this week we also had our regular monthly “books” call with our accountant and finance advisor. This was our usual opportunity to resolve any questions that have come up in relation to the accounts, but we also discussed progress on the improvements that we’ve been making. Infrastructure On the infrastructure side, the main highlight in recent weeks has been the migration from Anubis to Fastly’s Next-Gen Web Application Firewall (WAF) for protecting our infrastructure. The result of this migration will be an increased level of protection from bots, while simultaneously not interfering in peoples’ way when they’re using our infra. The Fastly product provides sophisticated detection of threats plus the ability for us to write our own fine-grained detection rules, so we can adjust firewall behaviour as we go. Huge thanks to Fastly for providing us with sponsorship for this service – it is a major improvement for our community and would not have been possible without their help. That’s it for this update. Thanks for reading and be on the lookout for the next update, probably in two weeks!

- This Week in GNOME: #241 Fifty! (2026/03/20 00:00)
  Update on what happened across the GNOME project in the week from March 13 to March 20. This week we released GNOME 50! This new major release of GNOME is full of exciting changes, including improved parental controls, many accessibility enhancements, expanded document annotation capabilities, calendar updates, and much more! See the GNOME 50 release notes and developer notes for more information. Readers who have been following this site will already be aware of some of the new features. If you’d like to follow the development of GNOME 51 (Fall 2026), keep an eye on this page - we’ll be posting exciting news every week! GNOME Circle Apps and Libraries gtk-rs ↗ Safe bindings to the Rust language for fundamental libraries from the GNOME stack. Julian reports I’ve added a chapter about accessibility to the gtk4-rs book. While I researched the topic beforehand and tested all examples with a screenreader, I would still appreciated additional feedback from people experienced with accessibility. Eyedropper ↗ Pick and format colors. FineFindus reports Eyedropper 2.2.0 is out now, bringing support for color picking without having the application open. It also now supports RGB in decimal notation and improves support for systems without a proper portal setup. As always, you can download the latest release from Flathub. Third Party Projects JumpLink announces The TypeScript type definitions generator ts-for-gir v4.0.0-beta.41 is out, and the big news is that we now have browsable API documentation for GJS TypeScript

bindings, live at https://gjsify.github.io/docs/. As a bonus, the same work also greatly improved the inline TypeScript documentation, hover docs in your editor are now much richer and more complete. Anton Isaiev reports RustConn is a GTK4/libadwaita connection manager for SSH, RDP, VNC, SPICE, Telnet, Serial, Kubernetes, and Zero Trust protocols. Core protocols use embedded Rust implementations - no external dependencies required. The 0.10.x series brings 8 new features and a major platform upgrade: New features: MOSH protocol support with predict mode, UDP port range, and server binary path Session recording in scriptreplay-compatible format with per-connection toggle and sensitive output sanitization Text highlighting rules - regex-based pattern matching with customizable colors, per-connection and global Ad-hoc broadcast - send keystrokes to multiple terminals simultaneously Smart Folders - dynamic connection grouping by protocol, tags, or host glob pattern Script credentials - resolve passwords from external commands with a Test button Per-connection terminal theming - background, foreground, and cursor color overrides CSV import/export with auto column mapping and configurable delimiter Platform changes: GTK-rs bindings upgraded to gtk4 0.11, libadwaita 0.9, vte4 0.10 Flatpak runtime bumped to GNOME 50 with VTE 0.80 Migrated to AdwSpinner, AdwShortcutsDialog, AdwSwitchRow, and AdwWrapBox (cfg-gated) FreeRDP 3.24.0 bundled in Flatpak - external RDP works out of the box on Wayland rdp file association - double-click to open and connect Split view now works with all VTE-based protocols 0.10.2 is a follow-up with 11 bug fixes for session recording, MOSH dispatch, highlight rules wiring, picocom detection in Flatpak, sidebar overflow, and RDP error messages. https://github.com/totoshko88/RustConn https://flathub.org/en/apps/io.github.totoshko88.RustConn Quadrapassel ↗ Fit falling blocks together. Will Warner reports Quadrapassel 50.0 has been released! This release has a lot of improvements for controls and polishes the app. Here is what's new: Made game view and preview exactly fit the blocks Improved game controller support Stopped duplicate keyboard events Replaced the libcanberra sound engine Fixed many small bugs and stylistic issues You can get Quadrapassel on Flathub. Documentation Jan-Willem says This week Java was added to the programming languages section on developer.gnome.org and many of the code examples were translated to Java. 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: Simple sort implementations vs production quality ones (2026/03/19 13:49)
  One of the most optimized algorithms in any standard library is sorting. It is used everywhere so it must be fast. Thousands upon thousands of developer hours have been sunk into inventing new algorithms and making sort implementations faster. Pystd has a different design philosophy where fast compilation times and readability of the implementation have higher priority than absolute performance. Perf still very much matters, it has to be fast, but not at the cost of 10x compilation time.This leads to the natural question of how much slower such an implementation would be compared to a production quality one. Could it even be faster? (Spoilers: no) The only way to find out is to run performance benchmarks on actual code.To keep things simple there is only one test set, sorting 10'000'000 consecutive 64 bit integers that have been shuffled to a random order which is the same for all algorithms. This is not an exhaustive test by any means but you have to start somewhere. All tests used GCC 15.2 using -O2 optimization. Pystd code was not thoroughly hand optimized, I only fixed (some of the) obvious hotspots.Stable sortPystd uses mergesort for stable sorting. The way the C++ standard specifies stable sort means that most implementations probably use it as well. I did not dive in the code to find out. Pystd's merge sort implementation consists of ~220 lines of code. It can be read on this page.Stdlibc++ can do the sort in 0.9 seconds whereas Pystd takes .94 seconds. Getting to within 5% with such a simple implementation is actually quite astonishing. Even when considering all the usual caveats where it might completely fall over with a different input data distribution and all that.Regular sortBoth stdlibc++ and Pystd use introsort. Pystd's implementation has ~150 lines of code but it also uses heapsort, which has a further 100 lines of code). Code for introsort is here, and heapsort is here.Stdlibc++ gets the sort done in 0.76 seconds whereas Pystd takes 0.82 seconds. This

makes it approximately 8% slower. It's not great, but getting within 10% with a few evening's work is still a pretty good result. Especially since, and I'm speculating here, std::sort has seen a lot more optimization work than std::stable_sort because it is used more.For heavy duty number crunching this would be way too slow. But for moderate data set sizes the performance difference might be insignificant for many use cases.Note that all of these are faster (note: did not measure) than libc's qsort because it requires an indirect function call on every comparison i.e. the comparison method can not be inlined.Compiling the stdlib version takes 0.68 seconds while the Pystd version takes 0.33 seconds. Without optimizations the times are 0.55 and 0.21 seconds, respectively.Where does the time go?Valgrind will tell you that quite easily.This picture shows quite clearly why big O notation can be misleading. Both quicksort (the inner loop of introsort) and heapsort have "the same" average time complexity but every call to heapsort takes approximately 4.5 times as long.

- Jakub Steiner: Friday Sketches (part 2) (2026/03/19 00:00)
  Two years have passed since I last shared my Friday app icon sketches, but the sketching itself hasn't stopped. For me, it's the best way to figure out the right metaphors before we move to final pixels. These sketches are just one part of the GNOME Design Team's wider effort to keep our icons consistent and meaningful—it is an endeavor that’s been going on for years. If you design a GNOME app following the GNOME Design Guidelines, feel free to request an icon to be made for you. If you are serious and apply for inclusion in GNOME Circle, you are way more likely to get a designer's attention. Previously
- Colin Walters: LLMs and core software: human driven (2026/03/18 20:17)
  It’s clear LLMs are one of the biggest changes in technology ever. The rate of progress is astounding: recently due to a configuration mistake I accidentally used Claude Sonnet 3.5 (released ~2 years ago) instead of Opus 4.6 for a task and looked at the output and thought “what is this garbage”? But daily now: Opus 4.6 is able to generate reasonable PoC level Rust code for complex tasks for me. It’s not perfect – it’s a combination of exhausting and exhilarating to find the 10% absolutely bonkers/broken code that still makes it past subagents. So yes I use LLMs every day, but I will be clear: if I could push a button to “un-invent” them I absolutely would because I think the long term issues in larger society (not being able to trust any media, and many of the things from Dario’s recent blog etc.) will outweigh the benefits. But since we can’t un-invent them: here’s my opinion on how they should be used. As a baseline, I agree with a lot from this doc from Oxide about LLMs. What I want to talk about is especially around some of the norms/tools that I see as important for LLM use, following principles similar to those. On framing: there’s “core” software vs “bespoke”. An entirely new capability of course is for e.g. a nontechnical restaurant owner to use an LLM to generate (“vibe code”) a website (excepting hopefully online orderings and payments!). I’m not overly concerned about this. Whereas “core” software is what organizations/businesses provide/maintain for others. I work for a company (Red Hat) that produces a lot of this. I am sure no one would want to run for real an operating system, cluster filesystem, web browser, monitoring system etc. that was primarily “vibe coded”. And while I respect people and groups that are trying to entirely ban LLM use, I don’t think that’s viable for at least my space. Hence the subject of this blog is my perspective on how LLMs should be used for “core” software: not vibe coding, but using LLMs responsibly and intelligently – and always under human control and review. Agents should amplify and be controlled by humans I think most of the industry would agree we can’t give responsibility to LLMs. That means they must be overseen by humans. If they’re overseen by a human, then I think they should be amplifying what that human thinks/does as a baseline – intersected with the constraints of the task of course. On “amplification”: Everyone using a LLM to generate content should inject their own system prompt (e.g. AGENTS.md) or equivalent. Here’s mine – notice I turn off all the emoji etc. and try hard to tune down bulleted lists because that’s not my style. This is a truly baseline thing to do. Now most LLM generated content targeted for

core software is still going to need review, but just ensuring that the baseline matches what the human does helps ensure alignment. Pull request reviews Let's focus on a very classic problem: pull request reviews. Many projects have wired up a flow such that when a PR comes in, it gets reviewed by a model automatically. Many projects and tools pitch this. We use one on some of my projects. But I want to get away from this because in my experience these reviews are a combination of: Extremely insightful and correct things (there's some amazing fine-tuning and tool use that must have happened to find some issues pointed out by some of these) Annoying nitpicks that no one cares about (not handling spaces in a filename in a shell script used for tests) Broken stuff like getting confused by things that happened after its training cutoff (e.g. Gemini especially seems to get confused by referencing the current date, and also is unaware of newer Rust features, etc) In practice, we just want the first of course. How I think it should work: A pull request comes in It gets auto-assigned to a human on the team for review A human contributing to that project is running their own agents (wherever: could be local or in the cloud) using their own configuration (but of course still honoring the project's default development setup and the project's AGENTS.md etc) A new containerized/sandboxed agent may be spawned automatically, or perhaps the human needs to click a button to do so – or perhaps the human sees the PR come in and thinks "this one needs a deeper review, didn't we hit a perf issue with the database before?" and adds that to a prompt for the agent. The agent prepares a draft review that only the human can see. The human reviews/edits the draft PR review, and has the opportunity to remove confabulations, add their own content etc. And to send the agent back to look more closely at some code (i.e. this part can be a loop) When the human is happy they click the "submit review" button. Goal: it is 100% clear what parts are LLM generated vs human generated for the reader. I wrote this agent skill to try to make this work well, and if you search you can see it in action in a few places, though I haven't truly tried to scale this up. I think the above matches the vision of LLMs amplifying humans. Code Generation There's no doubt that LLMs can be amazing code generators, and I use them every day for that. But for any "core" software I work on, I absolutely review all of the output – not just superficially, and changes to core algorithms very closely. At least in my experience the reality is still there's that percentage of the time when the agent decided to reimplement base64 encoding for no reason, or disable the tests claiming "the environment didn't support it" etc. And to me it's still a baseline for "core" software to require another human review to merge (per above!) with their own customized LLM assisting them (ideally a different model, etc). FOSS vs closed Of course, my position here is biased a bit by working on FOSS – I still very much believe in that, and working in a FOSS context can be quite different than working in a "closed environment" where a company/organization may reasonably want to (and be able to) apply uniform rules across a codebase. While for sure LLMs allow organizations to create their own Linux kernel filesystems or bespoke Kubernetes forks or virtual machine runtime or whatever – it's not clear to me that it is a good idea for most to do so. I think shared (FOSS) infrastructure that is productized by various companies, provided as a service and maintained by human experts in that problem domain still makes sense. And how we develop that matters a lot.

- Alberto Ruiz: Booting with Rust: Chapter 3 (2026/03/18 04:52)
  In Chapter 1 I gave the context for this project and in Chapter 2 I showed the bare minimum: an ELF that Open Firmware loads, a firmware service call, and an infinite loop. That was July 2024. Since then, the project has gone from that infinite loop to a bootloader that actually boots Linux kernels. This post covers the journey. The filesystem problem The Boot Loader Specification expects BLS snippets in a FAT filesystem under loaders/entries/. So the bootloader needs to parse partition tables, mount FAT, traverse directories, and read files. All #![no_std], all big-endian PowerPC. I tried writing my own minimal FAT32 implementation, then integrating simple-fatfs and fatfs. None worked well in a freestanding big-endian environment. Hadris The breakthrough was hadris, a no_std Rust crate supporting FAT12/16/32 and ISO9660. It needed some work to get

going on PowerPC though. I submitted fixes upstream for: thiserror pulling in std: default features were not disabled, preventing no_std builds. Endianness bug: the FAT table code read cluster entries as native-endian u32. On x86 that's invisible; on big-endian PowerPC it produced garbage cluster chains. Performance: every cluster lookup hit the firmware's block I/O separately. I implemented a 4MiB readahead cache for the FAT table, made the window size parametric at build time, and improved read_to_vec() to coalesce contiguous fragments into a single I/O. This made kernel loading practical. All patches were merged upstream. Disk I/O Hadris expects Read + Seek traits. I wrote a PROMDisk adapter that forwards to OF's read and seek client calls, and a Partition wrapper that restricts I/O to a byte range. The filesystem code has no idea it's talking to Open Firmware. Partition tables: GPT, MBR, and CHRP PowerVM with modern disks uses GPT (via the gpt-parser crate): a PReP partition for the bootloader and an ESP for kernels and BLS entries. Installation media uses MBR. I wrote a small mbr-parser subcrate using explicit-endian types so little-endian LBA fields decode correctly on big-endian hosts. It recognizes FAT32, FAT16, EFI ESP, and CHRP (type 0x96) partitions. The CHRP type is what CD/DVD boot uses on PowerPC. For ISO9660 I integrated hadris-iso with the same Read + Seek pattern. Boot strategy? Try GPT first, fall back to MBR, then try raw ISO9660 on the whole device (CD-ROM). This covers disk, USB, and optical media. The firmware allocator wall This cost me a lot of time. Open Firmware provides claim and release for memory allocation. My initial approach was to implement Rust's GlobalAlloc by calling claim for every allocation. This worked fine until I started doing real work: parsing partitions, mounting filesystems, building vectors, sorting strings. The allocation count went through the roof and the firmware started crashing. It turns out SLOF has a limited number of tracked allocations. Once you exhaust that internal table, claim either fails or silently corrupts state. There is no documented limit; you discover it when things break. The fix was to claim a single large region at startup (1/4 of physical RAM, clamped to 16-512 MB) and implement a free-list allocator on top of it with block splitting and coalescing. Getting this right was painful: the allocator handles arbitrary alignment, coalesces adjacent free blocks, and does all this without itself allocating. Early versions had coalescing bugs that caused crashes which were extremely hard to debug – no debugger, no backtrace, just writing strings to the OF console on a 32-bit big-endian target. And the kernel boots! March 7, 2026. The commit message says it all: "And the kernel boots!" The sequence: BLS discovery: walk loaders/entries/*.conf, parse into BLSEntry structs, filter by architecture (ppc64le), sort by version using rpmvercmp. ELF loading: parse the kernel ELF, iterate PT_LOAD segments, claim a contiguous region, copy segments to their virtual address offsets, zero BSS. Initrd: claim memory, load the initramfs. Bootargs: set /chosen/bootargs via setprop. Jump: inline assembly trampoline – r3=initrd address, r4=initrd size, r5=OF client interface, branch to kernel: core::arch::asm!("mr 7, 3", // save of_client "mr 0, 4", // r0 = kernel_entry "mr 3, 5", // r3 = initrd_addr "mr 4, 6", // r4 = initrd_size "mr 5, 7", // r5 = of_client "mtctr 0", "bctr", in("r3") of_client, in("r4") kernel_entry, in("r5") initrd_addr as usize, in("r6") initrd_size as usize, options(nostack, noreturn)) One gotcha: do NOT close stdout/stdin before jumping. On some firmware, closing them corrupts /chosen and the kernel hits a machine check. We also skip calling exit or release – the kernel gets its memory map from the device tree and avoids claimed regions naturally. The boot menu I implemented a GRUB-style interactive menu: Countdown: boots the default after 5 seconds unless interrupted. Arrow/PgUp/PgDn/Home/End navigation. ESC: type an entry number directly. e: edit the kernel command line with cursor navigation and word jumping (Ctrl+arrows). This runs on the OF console with ANSI escape sequences. Terminal size comes from OF's Forth interpret service (#columns / #lines), with serial forced to 80×24 because SLOF reports nonsensical values. Secure boot (initial, untested) IBM POWER has its own secure boot: the ibm,secure-boot device tree property (0=disabled, 1=audit, 2=enforce, 3=enforce+OS). The Linux kernel uses an appended signature format – PKCS#7 signed data appended to the kernel file, same format GRUB2 uses on IEEE 1275. I wrote an appended-sig crate that parses the appended signature layout, extracts an RSA key from a DER X.509 certificate (compiled in via include_bytes!), and verifies the signature (SHA-256/SHA-512) using the

RustCrypto crates, all no_std. The unit tests pass, including an end-to-end sign-and-verify test. But I have not tested this on real firmware yet. It needs a PowerVM LPAR with secure boot enforced and properly signed kernels, which QEMU/SLOF cannot emulate. High on my list. The ieee1275-rs crate The crate has grown well beyond Chapter 2. It now provides: claim/release, the custom heap allocator, device tree access (finddevice, getprop, instance-to-package), block I/O, console I/O with read_stdin, a Forth interpret interface, milliseconds for timing, and a GlobalAlloc implementation so Vec and String just work. Published on crates.io at github.com/rust-osdev/ieee1275-rs. What's next I would like to test the Secure Boot feature on an end to end setup but I have not gotten around to request access to a PowerVM PAR. Beyond that I want to refine the menu. Another idea would be to perhaps support the equivalent of the Unified Kernel Image using ELF. Who knows, if anybody finds this interesting let me know! The source is at the powerpc-bootloader repository. Contributions welcome, especially from anyone with POWER hardware access.

- Jakub Steiner: GNOME 50 Wallpapers (2026/03/18 00:00)
GNOME 50 just got released! To celebrate, I thought it'd be fun to look into the background (ding) of the newest additions to the collection. While the general aesthetic remains consistent, you might be surprised to see the default shifting from the long-standing triangular theme to hexagons. Well, maybe not that surprised if you've been following the gnome-backgrounds repo closely during the development cycle. We saw a rounded hexagon design surface back in 2024, but it was pulled after being deemed a bit too "flat" despite various lighting and color iterations. We've actually seen other hex designs pop up in 2020 and 2022, but they never quite got the ultimate spot as the default. Until now. Beyond the geometry shift of the default, the Symbolics wallpaper has also received its latest makeover. Truth be told, it hasn't historically been a fan favorite. I rarely see it in the wild in "show off your desktop" threads. Let's see if this new incarnation does any better. Similarly, the glass chip wallpaper has undergone a bit of a makeover as well. I'll also mention a... let's say less original design that caters to the dark theme folks out there. While every wallpaper in GNOME features a light and dark variant, Tubes comes with a dark and darker variant. I hope to see more of those "where did you get that wallpaper?" on reddit in 2026.
- Emmanuele Bassi: Let's talk about Moonforge (2026/03/17 17:45)
Last week, Igalia finally announced Moonforge, a project we've been working on for basically all of 2025. It's been quite the rollercoaster, and the announcement hit various news outlets, so I guess now is as good a time as any to talk a bit about what Moonforge is, its goal, and its constraints. Of course, as soon as somebody announces a new Linux-based OS, folks immediately think it's a new general purpose Linux distribution, as that's the square shaped hole where everything OS-related ends up. So, first things first, let's get a couple of things out of the way about Moonforge: Moonforge is not a general purpose Linux distribution Moonforge is not an embedded Linux distribution What is Moonforge Moonforge is a set of feature-based, well-maintained layers for Yocto, that allows you to assemble your own OS for embedded devices, or single-application environments, with specific emphasys on immutable, read-only root file system OS images that are easy to deploy and update, through tight integration with CI/CD pipelines. Why? Creating a whole new OS image out of whole cloth is not as hard as it used to be; on the desktop (and devices where you control the hardware), you can reasonably get away with using existing Linux distributions, filing off the serial numbers, and removing any extant packaging mechanism; or you can rely on the containerised tech stack, and boot into it. When it comes to embedded platforms, on the other hand, you're still very much working on bespoke, artisanal, locally sourced, organic operating systems. A good number of device manufacturers coalesced their BSPs around the Yocto Project and OpenEmbedded, which simplifies adaptations, but you're still supposed to build the thing mostly as a one off. While Yocto has improved leaps and bounds over the past 15 years, putting together an OS

image, especially when it comes to bundling features while keeping the overall size of the base image down, is still an exercise in artisanal knowledge. A little detour: Poky Twenty years ago, I moved to London to work for this little consultancy called OpenedHand. One of the projects that OpenedHand was working on was taking OpenEmbedded and providing a good set of defaults and layers, in order to create a "reference distribution" that would help people getting started with their own project. That reference was called Poky. We had a beaver mascot before it was cool These days, Poky exists as part of the Yocto Project, and it's still the reference distribution for it, but since it's part of Yocto, it has to abide to the basic constraint of the project: you still need to set up your OS using shell scripts and copy-pasting layers and recipes inside your own repository. The Yocto project is working on a setup tool to simplify those steps, but there are alternatives… Another little detour: Kas One alternative is kas, a tool that allows you to generate the local.conf configuration file used by bitbake through various YAML fragments exported by each layer you're interested in, as well as additional fragments that can be used to set up customised environments. Another feature of kas is that it can spin up the build environment inside a container, which simplifies enourmously its set up time. It avoids unadvertedly contaminating the build, and it makes it very easy to run the build on CI/CD pipelines that already rely on containers. What Moonforge provides Moonforge lets you create a new OS in minutes, selecting a series of features you care about from various available layers. Each layer provides a single feature, like: support for a specific architecture or device (QEMU x86_64, RaspberryPi) containerisation (through Docker or Podman) A/B updates (through RAUC, systemd-sysupdate, and more) graphical session, using Weston a WPE environment Every layer comes with its own kas fragment, which describes what the layer needs to add to the project configuration in order to function. Since every layer is isolated, we can reason about their dependencies and interactions, and we can combine them into a final, custom product. Through various tools, including kas, we can set up a Moonforge project that generates and validates OS images as the result of a CI/CD pipeline on platforms like GitLab, GitHub, and BitBucket; OS updates are also generated as part of that pipeline, just as comprehensive CVE reports and Software Bill of Materials (SBOM) through custom Yocto recipes. More importantly, Moonforge can act both as a reference when it comes to hardware enablement and support for BSPs; and as a reference when building applications that need to interact with specific features coming from a board. While this is the beginning of the project, it's already fairly usable; we are planning a lot more in this space, so keep an eye out on the repository. Trying Moonforge out If you want to check out Moonforge, I will point you in the direction of its tutorials, as well as the meta-derivative repository, which should give you a good overview on how Moonforge works, and how you can use it.

- Lucas Baudin: Improving Signatures in Papers: Malika's Outreachy Internship (2026/03/14 03:00)
  Last week was the end of Malika' internship within Papers about signatures that I had the pleasure to mentor. After a post about the first phase of Outreachy, here is the sequel of the story. Nowadays, people expect to be able to fill and sign PDF documents. We previously worked on features to insert text into documents and signatures needed to be improved. There is actually some ambiguity when speaking about signatures in PDFs: there are cryptographic signatures that guarantee that a certificate owner approved a document (now denoted by "digital" signatures) and there are also signatures that are just drawings on the document. These latter ones of course do not guarantee any authenticity but are more or less accepted in various situations, depending on the country. Moreover, getting a proper certificate to digitally sign documents may be complicated or costly (with the notable exception of a few countries providing them to their residents such as Spain). Papers lacked any support for this second category (that I will call "visual" signatures from now on). On the other hand, digital signing was implemented a few releases ago, but it heavily relies on Firefox certificate database 1 and in particular there is no way to manage personal certificates within Papers. During her three months internship, Malika implemented a new visual signatures management dialog and the corresponding UI to insert them, including nice

details such as image processing to import signature pictures properly. She also contributed to the poppler PDF rendering library to compress signature data. Then she looked into digital signatures and improved the insertion dialog, letting users choose visual signatures for them as well. If all goes well, all of this should be merged before Papers 51! Malika also implemented a prototype that allows users to import certificates and also deal with multiple NSS databases. While this needs more testing and code review2, it should significantly simplify digital signing. I would like to thank everyone who made this internship possible, and especially everyone who took the time to do calls and advise us during the internship. And of course, thanks to Malika for all the work she put into her internship! 1 or on NSS command line tools. 2 we don't have enough NSS experts, so help is very welcomed.

- Alice Mikhaylenko: Libadwaita 1.9 (2026/03/13 00:00)
  Another slow cycle, same as last time. Still, a few new things to showcase. Sidebars The most visible addition is the new sidebar widget. This is a bit confusing, because we already had widgets for creating windows with sidebars - AdwNavigationSplitView and AdwOverlaySplitView, but nothing to actually put into the sidebar pane. The usual recommendation is to build your own sidebar using GtkListBox or GtkListView, combined with the .navigation-sidebar style class. This isn't too difficult, but the result is zero consistency between different apps, not unlike what we had with GtkNotebook-based tabs in the past: It's even worse on mobile. In the best scenario it will just be a strangely styled flat list. Sometimes it will also have selection, and depending on how it's implemented it may be impossible to activate the selected row, like in libadwaita demo. So we have a pre-built one now. It doesn't aim to support every single use case (sidebars can get very complex, see e.g. GNOME Builder), but just to be good enough for the basic situations. How basic is basic? Well, it has selection, sections (with or without titles), tooltips, context menus, a drop target, suffix widgets at the end of each item's row, auto-activation when hovered during drag-n-drop. A more advanced feature is built-in search filter - via providing a GtkFilter and a placeholder page. And that's about it. There will likely be more features in future, like collapsible sections and drag source on items, rather than just a drop target, but this should already be enough for quite a lot of apps. Not everything, but that's not the goal here. Internally, it's using GtkListBox. This means that it doesn't scale to thousands of items the way GtkListView would, but we can have much tighter API and mobile integration. Now, let's talk about mobile. Ideally sidebars on mobile wouldn't really be sidebars at all. This pattern inherently requires a second pane, and falls apart otherwise. AdwNavigationSplitView already presents the sidebar pane as a regular page, so let's go further and turn sidebars into boxed lists. We're already using GtkListBox, after all. So - AdwSidebar has the mode property. When set to ADW_SIDEBAR_MODE_PAGE, it becomes a page of boxed lists - indistinguishable from any others. It hides item selection, but it's still tracked internally. It can still be changed programmatically, and changes when an item is activated. Once the sidebar mode is set back to ADW_SIDEBAR_MODE_SIDEBAR, it will reappear. Internally it's nothing special, as it just presents the same data using different widgets. The adaptive layouts page has a detailed example for how to create UIs like this, as well as the newly added section about overlay sidebars that don't change as drastically. View switcher sidebar Once we have a sidebar, a rather obvious thing to do is to provide a GtkStackSidebar replacement. So AdwViewSwitcherSidebar is exactly that. It works with AdwViewStack rather than GtkStack, and has all the same features as existing view switcher, as well as an extra one - sections. To support that, AdwViewStackPage has new API for defining sections - the :starts-section and :section-title properties, while the AdwViewStack:pages) model is now a section model. Like regular sidebars, it supports the boxed list mode and search filtering. Unlike other view switchers or GtkStackSidebar, it also exposes AdwSidebar's item activation signal. This is required to make it work on mobile. Demo improvements The lack of sidebar was the main blocker for improving libadwaita demo in the past. Now that it's solved, the demo is at last, fully adaptive. The sidebar has been reorganized into sections, and has icons and search now. This also unblocks other

potential improvements, such as having a more scalable preferences dialog. Reduced motion While there isn't any new API, most widgets with animations have been updated to respect the new reduced motion preference - mostly by replacing sliding/scaling animations with crossfades, or otherwise toning down animations when it's impossible: AdwDialog open/close transitions are crossfades except for the swipe-to-close gesture AdwBottomSheet transition is a crossfade when there's no bottom bar, and a slide without overshooting if there is AdwNavigationView transition is a crossfade except when using the swipe gestures AdwTabOverview transition is a crossfade AdwOverlaySplitView is unaffected for now. Same for toasts, those are likely small enough to not cause motion sickness. If it turns out to be a problem, it can be changed later. I also didn't update any of the deprecated widgets, like AdwLeaflet. Applications still using those should switch to the modern alternatives. The prefers-reduced-motion media feature is available for use from app CSS as well, following the GTK addition. Other changes AdwAboutDialog rows that contain links have a context menu now. Link rows may become a public widget in future if there's interest. GTK_DEBUG=builder diagnostics are now supported for all libadwaita widgets. This can be used to find places where <child> tags are used in UI when equivalent properties exist. Following GTK, all GListModel implementations now come with :item-type and :n-item properties, to make it easier to use them from expressions. The AdwTabView:pages model implements sections now: one for pinned pages and one for everything else. AdwToggle has a new :description property that can be used to set accessible description for individual toggles separately from tooltips. Adrien Plazas improved accessibility in a bunch of widgets. The majority of this work has been backported to 1.8.x as well. For example, AdwViewSwitcher and AdwInlineViewSwither now read out number badges and needs attention status. AdwNoneAnimationTarget now exists for situations where animations are used as frame clock-based timers, as an alternative to using AdwCallbackAnimationTarget with empty callback. AdwPreferencesPage will refuse to add children of types other than AdwPreferencesGroup, instead of overlaying them over the page and then leaking them after the page is destroyed. This change was backported to 1.8.2 and subsequently reverted in 1.8.3 as it turned out multiple apps were relying on the broken behavior. Maximiliano made non-nullable string setter functions automatically replace NULL parameters with empty strings, since allowing NULL breaks Rust bindings, while rejecting them means apps using expressions get unexpected criticals - for example, when accessing a non-nullable string property on an object, and that object itself is NULL. As mentioned in the 1.8 blog post, style-dark.css, style-hc.css and style-hc-dark.css resources are now deprecated and apps using them will get warnings on startup. Apps are encouraged to switch to a single style.css and conditionally load styles using media queries instead. While not a user-visible change (hopefully!), the internal stylesheet has been refactored to use prefers-contrast media queries for high contrast styles instead of 2 conditionally loaded variants - further reducing the need on SCSS, even if not entirely replacing it just yet. (the main blocker is @extend, as well nesting and a few mixins, such as focus ring) Future A big change in works is a revamp of icon API. GTK has a new icon format that supports stateful icons with animated transitions, variable stroke weight, and many other capabilities. Currently, libadwaita doesn't make use of this, but it will in future. In fact, a few smaller changes are already in 1.9: all of the internal icons in libadwaita itself, as well as in the demo and docs, have been updated to use the new format. Thanks to the GNOME Foundation for their support and thanks to all the contributors who made this release possible. Because 2026 is such an interesting period of time to live in, I feel I should explicitly say that libadwaita does not contain any AI slop, nor does allow such contributions, nor do I have any plans to change that. Same goes for all of my other projects, including this website.

- This Week in GNOME: #240 Big Reworks (2026/03/13 00:00)
  Update on what happened across the GNOME project in the week from March 06 to March 13. GNOME Core Apps and Libraries Files ↗ Providing a simple and integrated way of managing your files and browsing your file system. Peter Eisenmann announces For version 50 Files aka nautilus

has retrieved many bug fixes, tiny niceties and big reworks. The most prominent are: Faster thumbnail and icon loading Pop-out property dialogs for free-floating windows Reworked batch rename mechanism and highligths for replaced text Shorter file operation descriptions in sidebar Support for multiple simulatenous file type search filters Case-insensitive pathbar completions Dedicated dialog for grid view captions Reduced memory usage Internal modernizations including usage of Blueprint and glycin Increased test coverage (23% → 37%) A big thank you to all contributing coders and translators! 📄 Document Viewer (Papers) ↗ View, search or annotate documents in many different formats. lbaudin says Malika's Outreachy internship just ended! If all goes well, her work on improving signatures in Papers should land during next cycle. Read more about it here. Libadwaita ↗ Building blocks for modern GNOME apps using GTK4. Alice (she/her) 🏳️‍⚧️ says I released libadwaita 1.9! Read the accompanying blog post to see what's new Third Party Projects Haydn Trowell says Typesetter, the minimalist Typst editor, now speaks more languages. With the latest update, you can now use it in Chinese, French, Spanish, Turkish, and German. Thanks to Dawn Chan, Philippe Charlanes, XanderLeaDaren, Roger Weissenbrunner, Sabri Ünal, and Sebastian Kern for their time and effort! Get in on Flathub: https://flathub.org/apps/net.trowell.typesetter If you want to help bring Typesetter to your language, translations can be contributed via Weblate (https://translate.codeberg.org/engage/typesetter/). Anton Isaiev announces I am incredibly excited to share the latest news about RustConn, covering the massive journey from version 0.9.4 to 0.9.15! This release cycle focused on making the app's internal architecture as robust as its features. During this time, we closed dozens of feature requests and fixed numerous critical bugs. Here are the most important improvements from the recent updates: Flawless Flatpak Experience: I completely resolved issues with importing Remmina configurations inside the sandbox and fixed specific SSH password prompt display bugs in environments like KDE. Memory-Level Security: I introduced strict zeroing of Bitwarden master passwords in memory immediately after use. Additionally, I completely dropped the external sshpass dependency to enhance overall security. Advanced Connections: The native SPICE client is now enabled by default. For RDP sessions, I added a convenient "Quick Actions" menu (one-click access to Task Manager, PowerShell, etc.), and for VNC, I introduced flexible encoding options. Code & UI Cleanup: I completed a major refactoring of the UI modules (some became 5x lighter!), which eliminated text-clipping issues in dialogs and significantly improved application performance. I want to express a huge thank you to everyone who uses RustConn and takes the time to provide feedback! Your positive reviews and comments are the main thing that motivates me to work on the project every single day. At the same time, your bug reports and feature ideas are exactly what make these releases possible. Thank you for being such an amazing community! https://github.com/totoshko88/RustConn https://flathub.org/en/apps/io.github.totoshko88.RustConn Mikhail Kostin announces Vinyl is a new (one more :D) music player. Vinyl built on rust with relm4. The first stable version already available on Flathub and provides features: Simple user-friendly interface inspired by amberol. Basic media controls. Lyrics (.lrc) support MPRIS support for controlling Vinyl from other applications. Save playlist and track/position of track, that played before the app close Gir.Core ↗ Gir.Core is a project which aims to provide C# bindings for different GObject based libraries. Marcel Tiede reports GirCore released new C# bindings in version 0.8.0-preview.1. It incldues new GTK composite template support and added bindings for GdkWayland-4.0. Miscellaneous GNOME OS ↗ The GNOME operating system, development and testing platform Valentin David announces GNOME OS now has kmscon enabled by default. Kmscon is a KMS/DRM userspace terminal that replaces the Linux virtual terminals (the ones from ctrl-alt-f#). It is a lot more configurable. So next time you try to debug GNOME Shell from a virtual terminal and the font is too small, press "ctrl +". 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!

- Aryan Kaushik: Open Forms is now 0.4.0 - and the GUI Builder is here (2026/03/12 20:51)
  Open Forms is now 0.4.0 - and the GUI Builder is here A quick recap for the newcomers Ever been to a conference where you set up a booth or

tried to collect quick feedback and experienced the joy of: Captive portal logout Timeouts Flaky Wi-Fi drivers on Linux devices Poor bandwidth or dead zones This is exactly what happened while setting up a booth at GUADEC. The Wi-Fi on the Linux tablet worked, we logged into the captive portal, the chip failed, Wi-Fi gone. Restart. Repeat. We eventually worked around it with a phone hotspot, but that locked the phone to the booth. A one-off inconvenience? Maybe. But at any conference, summit, or community event, at least one of these happens reliably. So I looked for a native, offline form collection tool. Nothing existed without a web dependency. So I built one. Open Forms is a native GNOME app that collects form inputs locally, stores responses in CSV, works completely offline, and never touches an external service. Your data stays on your device. Full stop. What's new in 0.4.0 - the GUI Form Builder The original version shipped with one acknowledged limitation: you had to write JSON configs by hand to define your forms. Now, I know what you're thinking. "Writing JSON to set up a form? That's totally normal and not at all a terrible first impression for non-technical users." And you'd be completely wrong, to me it was normal and then my sis had this to say "who even thought JSON for such a basic thing is a good idea, who'd even write one" which was true. I knew it and hence it was always on the roadmap to fix, which 0.4.0 finally fixes. Open Forms now ships a full visual form builder. Design a form entirely from the UI - add fields, set labels, reorder things, tweak options, and hit Save. That's it. The builder writes a standard JSON config to disk, same schema as always, so nothing downstream changes. It also works as an editor. Open an existing config, click Edit, and the whole form loads up ready to tweak. Save goes back to the original file. No more JSON editing required. Libadwaita is genuinely great The builder needed to work well on both a regular desktop and a Linux phone without me maintaining two separate layouts or sprinkling breakpoints everywhere. Libadwaita just... handles that. The result is that Open Forms feels native on GNOME and equally at home on a Linux phone, and I genuinely didn't have to think hard about either. That's the kind of toolkit win that's hard to overstate when you're building something solo over weekends. The JSON schema is unchanged If you already have configs, they work exactly as before. The builder is purely additive, it reads and writes the same format. If you like editing JSON directly, nothing stops you. I'm not going to judge, but my sister might. Also thanks to Felipe and all others who gave great ideas about increasing maintainability. JSON might become a technical debt in future, and I appreciate the insights about the same. Let's see how it goes. Install Snap Store snap install open-forms Flatpak / Build from source See the GitHub repository for build instructions. There is also a Flatpak release available. What's next A11y improvements Maybe and just maybe an optional sync feature Hosting on Flathub - if you've been through that process and have advice, please reach out Open Forms is still a small, focused project doing one thing. If you've ever dealt with Wi-Fi pain while collecting data at an event, give it a try. Bug reports, feature requests, and feedback are all very welcome. And if you find it useful - a star on GitHub goes a long way for a solo project. □ Open Forms on GitHub

- Sebastian Wick: Redefining Content Updates in Wayland (2026/03/10 22:56)
  The Wayland core protocol has described surface state updates the same way since the beginning: requests modify pending state, commits either apply that state immediately or cache it into the parent for synchronized subsurfaces. Compositors implemented this model faithfully. Then things changed. Buffer Readiness and Compositor Deviation The problem emerged from GPU work timing. When a client commits a surface with a buffer, that buffer might still have GPU rendering in progress. If the compositor applies the commit immediately, it would display incomplete content—glitches. If the compositor submits its own GPU work with a dependency on the unfinished client work, it risks missing the deadlines for the next display refresh cycles and even worse stalling in some edge cases. To get predictable timing, the compositor needs to defer applying commits until the GPU work finishes. This requires tracking readiness constraints on committed state. Mutter was the first compositor to address this by implementing constraints and dependency tracking of content updates internally. Instead of immediately applying

or caching commits, Mutter queued the changes in what we now call content updates, and only applied them when ready. Critically, this was an internal implementation detail. From the client’s perspective, the protocol semantics remained unchanged. Mutter had deviated from the implementation model implied by the specification while maintaining the observable behavior. New Protocols on Unstable Foundations When we wanted better frame timing control and a proper FIFO presentation modes on Wayland, we suddenly required explicit queuing of content updates to describe the behavior of the protocols. You can’t implement FIFO and scheduling of content updates without a queue, so both the fifo and commit-timing protocols were designed around the assumption that compositors maintain per-surface queues of content updates. These protocols were implemented in compositors on top of their internal queue-based architectures, and added to wayland-protocols. But the core protocol specification was never updated. It still described the old “apply or cache into parent state” model that has no notion of content updates, and per-surface queues. We now had a situation where the core protocol described one model, extension protocols assumed a different model, and compositors implemented something that sort of bridged both. Implementation and Theory That situation is not ideal: If the internal implementation follows the design which the core protocol implies, you can’t deal properly with pending client GPU work, and you can’t properly implement the latest timing protocols. To understand and implement the per-surface queue model, you would have to read a whole bunch of discussions, and most likely an implementation such as the one in mutter. The implementations in compositors also evolved organically, making them more complex than they actually have to be. To make matter worse, we also lacked a shared vocabulary for discussing the behavior. The obvious solution to this is specifying a general model of the per-surface content update queues in the core protocol. Easier said than done though. Coming up with a model that is sufficient to describe the new behavior while also being compatible with the old behavior when no constraints on content updates defer their application was harder than I expected. Together with Julian Orth, we managed to change the Wayland core protocol, and I wrote documentation about the system. Recently Pekka Paalanen and Julian Orth reviewed the work, which allowed it to land. The updated and improved Wayland book should get deployed soon, as well. The end result is that if you ever have to write a Wayland compositor, one of the trickier parts to get right should now be almost trivial. Implement the rules as specified, and things should just work. Edge cases are handled by the general rules rather than requiring special knowledge.

- Andy Wingo: nominal types in webassembly (2026/03/10 08:19)
  Before the managed data types extension to WebAssembly was incorporated in the standard, there was a huge debate about type equality. The end result is that if you have two types in a Wasm module that look the same, like this:(type $t (struct i32)) (type $u (struct i32)) Then they are for all intents and purposes equivalent. When a Wasm implementation loads up a module, it has to partition the module’s types into equivalence classes. When the Wasm program references a given type by name, as in (struct.get $t 0) which would get the first field of type $t, it maps $t to the equivalence class containing $t and $u. See the spec, for more details.This is a form of structural type equality. Sometimes this is what you want. But not always! Sometimes you want nominal types, in which no type declaration is equivalent to any other. WebAssembly doesn’t have that, but it has something close: recursive type groups. In fact, the type declarations above are equivalent to these:(rec (type $t (struct i32))) (rec (type $u (struct i32))) Which is to say, each type is in a group containing just itself. One thing that this allows is self-recursion, as in:(type $succ (struct (ref null $succ))) Here the struct’s field is itself a reference to a $succ struct, or null (because it’s ref null and not just ref).To allow for mutual recursion between types, you put them in the same rec group, instead of each having its own:(rec (type $t (struct i32)) (type $u (struct i32))) Between $t and $u we don’t have mutual recursion though, so why bother? Well rec groups have another role, which is that they are the unit of structural type equivalence. In this case, types $t and $u are not in the same equivalence class, because they are part of the same rec

group. Again, see the spec.Within a Wasm module, rec gives you an approximation of nominal typing. But what about between modules? Let’s imagine that $t carries important capabilities, and you don’t want another module to be able to forge those capabilities. In this case, rec is not enough: the other module could define an equivalent rec group, construct a $t, and pass it to our module; because of isorecursive type equality, this would work just fine. What to do?cursèd nominal typingI said before that Wasm doesn’t have nominal types. That was true in the past, but no more! The nominal typing proposal was incorporated in the standard last July. Its vocabulary is a bit odd, though. You have to define your data types with the tag keyword:(tag $v (param $secret i32)) Syntactically, these data types are a bit odd: you have to declare fields using param instead of field and you don’t have to wrap the fields in struct.They also omit some features relative to isorecursive structs, namely subtyping and mutability. However, sometimes subtyping is not necessary, and one can always assignment-convert mutable fields, wrapping them in mutable structs as needed.To construct a nominally-typed value, the mechanics are somewhat involved; instead of (struct.new $t (i32.const 42)), you use throw:(block $b (result (ref exn)) (try_table (catch_all_ref $b) (throw $v (i32.const 42))) (unreachable)) Of course, as this is a new proposal, we don’t yet have precise type information on the Wasm side; the new instance instead is returned as the top type for nominally-typed values, exn.To check if a value is a $v, you need to write a bit of code:(func $is-v? (param $x (ref exn)) (result i32) (block $yep (result (ref exn)) (block $nope (try_table (catch_ref $v $yep) (catch_all $nope) (throw_ref (local.get $x)))) (return (i32.const 0))) (return (i32.const 1))) Finally, field access is a bit odd; unlike structs which have struct.get, nominal types receive all their values via a catch handler.(func $v-fields (param $x (ref exn)) (result i32) (try_table (catch $v 0) (throw_ref (local.get $x))) (unreachable)) Here, the 0 in the (catch $v 0) refers to the function call itself: all fields of $v get returned from the function call. In this case there’s only one, othewise a get-fields function would return multiple values. Happily, this accessor preserves type safety: if $x is not actually $v, an exception will be thrown.Now, sometimes you want to be quite strict about your nominal type identities; in that case, just define your tag in a module and don’t export it. But if you want to enable composition in a principled way, not just subject to the randomness of whether another module happens to implement a type structurally the same as your own, the nominal typing proposal also gives a preview of type imports. The facility is direct: you simply export your tag from your module, and allow other modules to import it. Everything will work as expected!finFriends, as I am sure is abundantly clear, this is a troll post :) It’s not wrong, though! All of the facilities for nominally-typed structs without subtyping or field mutability are present in the exception-handling proposal.The context for this work was that I was updating Hoot to use the newer version of Wasm exception handling, instead of the pre-standardization version. It was a nice change, but as it introduces the exnref type, it does open the door to some funny shenanigans, and I find it hilarious that the committee has been hemming and hawwing about type imports for 7 years and then goes and ships it in this backward kind of way.Next up, exception support in Wastrel, as soon as I can figure out where to allocate type tags for this new nominal typing facility. Onwards and upwards!

- Andy Wingo: free trade and the left, ter: mises and my apostasy (2026/03/06 21:05)
  Good evening. Let’s talk about free trade!Last time, we discussed Marc-William Palen’s Pax Economica, which looks at how the cause of free trade was taken up by a motley crew of anti-imperialists, internationalists, pacifists, marxists, and classical liberals in the nineteenth century. Protectionism was the prerogative of empire—only available to those with a navy—and it so it makes sense that idealists might support “peace through trade”. So how did free trade go from a cause of the “another world is possible” crowd to the halls of the WTO? Did we leftists catch a case of buyer’s remorse, or did the goods delivered simply not correspond to the order?To make an attempt at an answer, we need more history. From the acknowledgements of Quinn Slobodian’s Globalists: This book is a long-simmering product of the Seattle protests against the World Trade Organization in 1999. I was part of a generation that came of age after the Cold War's end. We became adolescents in the midst of talk of

globalization and the End of History. In the more hyperactive versions of this talk, we were made to think that nations were over and the one indisputable bond uniting humanity was the global economy. Seattle was a moment when we started to make collective sense of what was going on and take back the story line. I did not make the trip north from Portland but many of my friends and acquaintances did, painting giant papier-mâché fists red to strap to backpacks and coming back with takes of zip ties and pepper spray, nights in jail, and encounters with police—tales they spun into war stories and theses. This book is an apology for not being there and an attempt to rediscover in words what the concept was that they went there to fight. Slobodian's approach is to pull on the thread that centers around the WTO itself. He ends up identifying what he calls the "Geneva School" of neoliberalism: from Mise's circle in Vienna, to the International Chamber of Commerce in Paris, to the Hayek-inspired Mont Pèlerin Society, to Petersmann of the WTO precursor GATT organization, Röpke of the Geneva Graduate Institute of International Studies, and their lesser successors of the 1970s and 1980s.The thesis that Slobodian ends up drawing is that neoliberalism is not actually a laissez-faire fundamentalism, but rather an ideology that placed the value of free-flowing commerce above everything else: above democracy, above sovereignty, above peace, and that as such it actually requires active instutional design to protect commerce from the dangers of, say, hard-won gains by working people in one country (Austria, 1927), expropriation of foreign-owned plantations in favor of landless peasants (Guatemala, 1952), internal redistribution within countries transitioning out of minority rule (South Africa, 1996), decolonization (1945-1975 or so), or just the election of a moderate socialist at the ballot box (Chile, 1971).Now, dear reader, I admit to the conceit that if you are reading this, probably you are a leftist also, and if not, at least you are interested in understanding how it is that we think, with what baubles do we populate our mental attics, that sort of thing. Well, friend, you know that by the time we get to Chile and Allende we are stomping and clapping our hands and shouting in an extasy of indignant sectarian righteousness. And that therefore should we invoke the spectre of neoliberalism, it is with the deepest of disgust and disdain: this project and all it stands for is against me and mine. I hate it like I hated Henry Kissinger, which is to say, a lot, viscerally, it hurts now to think of it, rest in piss you bastard.two theologiesAnd yet, I'm still left wondering what became of the odd alliance of Marx with Manchester liberalism. Palen's Pax Economica continues to sketch a thin line through the twentieth century, focusing on showing the continued presence of commercial-peace exponents despite it not turning out to be our century. But the rightward turn of the main contingent of free-trade supporters is not explained. I have an idea about how it is that this happened; it is anything but scholarly, but here we go.Let us take out our coarsest brush to paint a crude story: the 19th century begins in the wake of the American and French revolutions, making the third estate and the bourgeoisie together the revolutionary actors of history. It was a time in which "we" could imagine organizing society in different ways, the age of the utopian imaginary, but overlaid with the structures of the old, old money, old land ownership, revanchist monarchs, old power, old empire. In this context, Cobden's Anti-Corn Law League was insurgent, heterodox, asking for a specific political change with the goal of making life on earth better for the masses. Free trade was a means to an end. Not all Cobdenites had the same ends, but Marx and Manchester both did have ends, and they happened to coincide in the means.Come the close of the Great War in 1918, times have changed. The bourgeoisie have replaced the nobility as the incumbent power, and those erstwhile bourgeois campaigners now have to choose between idealism and their own interest. But how to choose?Some bourgeois campaigners will choose a kind of humanist notion of progress; this is the thread traced by Palen, through the Carnegie Endowment for International Peace, the Young Women's Christian Association, the Haslemere Group, and others.Some actors are not part of the hegemonic bourgeoisie at all, and so have other interests. The newly independent nations after decolonization have more motive to upend the system than to preserve it; their approach to free trade has both tactical and ideological components. Tactical, in the sense that they wanted access to first-world markets, but also sometimes some protections for their own industries;

ideological, in the sense that they often acted in solidarity with other new nations against the dominant powers. In addition to the new nations, the Soviet bloc had its own semi-imperial project, and its own specific set of external threats; we cannot blame them for being tactical either.And then you have Ludwig von Mises. Slobodian hints at Mises' youth in the Austro-Hungarian empire, a vast domain of many languages and peoples but united by trade and the order imposed by monarchy. After the war and the breakup of the empire, I can only imagine—and here I am imagining, this is not a well-evidenced conclusion—I imagine he felt a sense of loss. In the inter-war, he holds court as the doyen of the Vienna Chamber of Commerce, trying to put the puzzle pieces back together, to reconstruct the total integration of imperial commerce, but from within Red Vienna. When in 1927, a court decision acquitted a fascist milicia that fired into a crowd, killing a worker and a child, the city went on general strike, and workers burned down the ministry of justice. Police responded violently, killing 89 people and injuring over 1000. Mises was delighted: order was restored.And now, a parenthesis. I grew up Catholic, in a ordinary kind of way. Then in my early teens, I concluded that if faith meant anything, it has to burn with a kind of fervor; I became an evangelical Catholic, if such is a thing. There were special camps you could go to with intense emotional experiences and people singing together and all of that is God, did you know? Did you know? The feelings attenuated over time but I am a finisher, and so I got confirmed towards the end of high school. I went off to university for physics and stuff and eventually, painfully, agonizingly concluded there was no space for God in the equations.Losing God was incredibly traumatic for me. Not that I missed, like, the idea of some guy, but as someone who wants things to make sense, to have meaning, to be based on something, anything at all: losing a core value or morality invalidated so many ideas I had about the world and about myself. What is the good life, a life well led? What is true and right in a way that is not contingent on history? I am embarrassed to say that for a while I took the UN declaration of human rights to be axiomatic.When I think about Mise's reaction to the 1927 general strike in Vienna, I think about how I scrambled to find something, anything, to replace my faith in God. As the space for God shrank with every advance in science, some chose to identify God with his works, and then to progressively ascribe divine qualities to those works: perhaps commerce is axiomatically Good, and yet ineffable, in the sense that it is Good on its own, and that no mortal act can improve upon it. How else can we interpret Hayek's relationship with the market except as awe in the presence of the divine?This is how I have come to understand the neoliberal value system: a monotheism with mammon as godhead. There may be different schools within it, but all of the faithful worship the same when they have to choose between, say, commerce and democracy, commerce and worker's rights, commerce and environmental regulation, commerce and taxation, commerce and opposition to apartheid. It's a weird choice of deity. Now that God is dead, one could have chosen anything to take His place, and these guys chose the "global economy". I would pity them if I still had a proper Christian heart.means without endI think that neoliberals made a miscalculation when they concluded that the peace of doux commerce is not predicated on justice. Sure, in the short run, you can do business with Pinochet's Chile, privatize the national mining companies, and cut unemployment benefits, but not without incurring moral damage; people will see through it, in time, as they did in Seattle in 1999. Slobodian refers to the ratification of the WTO as a Pyrrhic victory; in their triumph, neoliberals painted a target on their backs.Where does this leave us now? And what about Mercosur? I'm starting to feel the shape of an answer, but I'm not there yet. I think we'll cover the gap between Seattle and the present day in a future dispatch. Until then, let's take care of one other; as spoke the prophet Pratchett, there's no justice, just us.

- Allan Day: GNOME Foundation Update, 2026-03-06 (2026/03/06 18:38)
  This post is the latest in my series of GNOME Foundation updates. I'm writing these in my capacity as Foundation President, where I'm busy managing a lot of what's happening at the organisation at the moment. Each of these posts is a report on what happened over a particular

period, and this post covers the current week as well as the previous one (23rd February to 6th March). Audit time I've mentioned the GNOME Foundation's audit on numerous occassions previously. This is being conducted as a matter of routine, but it is our first full formal audit, so we have been learning a lot about what's involved. This week has been the audit fieldwork itself, which has been quite intense and a lot of work for everyone involved. The audit team consists of 5 people, most of whom are accountants of different grades. Our own finance team has been meeting with them three times a day since Tuesday, answering questions, doing walkthroughs of our systems, and providing additional documents as requested. A big part of the audit is cross-referencing and checking documentation, and we have been busy responding to requests for information throughout the week. On last count, we have provided 140 documents to the auditors this week alone, on 20 different themes, including statements, receipts, contracts, invoices, sponsorship agreements, finance reports, and so on. We're expecting the draft audit report in about three weeks. Initial signs are good! GUADEC 2026 Planning activity for GUADEC 2026 has continued over the past two weeks. That includes organising catering, audio visual facilities, a photographer, and sponsorship work. Registration for the event is now open. The Call for Papers is also open and will close on 13 March – just one week away! If you would like to present this year, please submit an abstract! If you would like travel sponsorship for GUADEC, there are two deadlines to submit a request: 15th March (for those who need to book travel early, such as if they need a visa) and 24th May (for those with less time pressure). LAS 2026 This year's Linux App Summit is happening in Berlin, on the 16th and 17th May, and is shaping up to be a great event. As usual we are co-organizing the event with KDE, and the call for proposals has just opened. If you'd like to present, you have until 23rd March to submit a paper. The Travel Committee will be accepting travel applications for LAS attendees this year, so if you'd like to attend and need travel assistance, please submit a request no later than 13th April. Infrastructure On the infrastructure side, GNOME's single sign on service has been integrated with blogs.gnome.org, which is great for security, as well as meaning that you won't need to remember an extra password for our WordPress instance. Many thanks to miniOrange for providing us with support for their OAuth plugin for WordPress, which has allowed this to happen! That's it for my update this week. In addition to the highlights that I've mentioned, there are quite a number of other activities happening at the Foundation right now, particularly around new programs, some of which we're not quite ready to talk about, but hope to provide updates on soon.

- Sophie Herold: What you might want to know about painkillers (2026/03/04 19:22)
  Painkillers are essential. (There are indicators that Neanderthals already used them.) However, many people don't know about aspects of them, that could be relevant for them in practice. Since I learned some new things recently, here a condensed info dump about painkillers. Many aspects here are oversimplified in the hope to raise some initial awareness. Please consult your doctor or pharmacist about your personal situation, if that's possible. I will not talk about opioids. Their addiction potential should never be underestimated. Here is the short summary: Find out which substance and dose works for you. With most painkillers, check if you need to take Pantoprazole to protect your stomach. Never overdose paracetamol, never take it with alcohol. If possible, take pain medication early and directly in the dose you need. Don't take pain medication for more than 15 days a month against headaches. Some mediaction even fewer days. If you have any preexisting conditions, health risks, or take additional medication, check very carefully if any of these things could interacts with your pain medication. Not all substances will work for you The likelihood of some substances not working for some sort of pain for you is pretty high. If something doesn't seem to work for you, consider trying a different substance. I have seen many doctors being very confident that a substance must work. The statistics often contradict them. Common over the counter options are: Ibuprofen Paracetamol Naproxen Acetylsalicylic Acid (ASS) Diclofenac All of them also reduce fever. All of them, except Paracetamol, are anti-inflammatory. The anti-inflammatory effect is highest in Diclofenac and Naproxen, still

significant in Ibuprofen. It might very well be that none of them work for you. In that case, there might still be other options to prevent or treat your pain. Gastrointestinal (GI) side effects All nonsteroidal anti-inflammatory drugs (NSAIDs), that is, Ibuprofen, Naproxen, ASS, and, Diclofenac can be hard on your stomach. This can be somewhat mitigated by taking them after a meal and with a lot of water. Among the risk factors you should be aware of are Age above 60, history of GI issues, intake of an SSRI, SNRI, or Steroids, consumption of alcohol, or smoking. The risk is lower with Ibuprofen, but higher for ASS, Naproxen, and, especially, Diclofenac. It is common to mitigate the GI risks by taking a Proton Pump Inhibitor (PPI) like Pantoprazole 20 mg. Usually, if any of the risk factors apply to you. You can limit the intake to the days where you use painkillers. You only need one dose per day, 30–60 minutes before a meal. Then you can take the first painkiller for the day after the meal. Taking Pantoprazole for a few days a month is usually fine. If you need to take it continuously or very often, you have to very carefully weigh all the side effects of PPIs. Paracetamol doesn’t have the same GI risks. If it is effective for you, it can be an option to use it instead. It is also an option to take a lower dose NSAIDs and a lower dose of paracetamol to minimize the risks of both. Metamizole is also a potential alternative. It might, however, not be available in your country, due to a rare severe side effect. If available, it is still a potential option in cases where other side effects can also become very dangerous. It is usually prescription-only. For headaches, you might want to look into Triptans. They are also usually prescription-only. Liver related side effects Paracetamol can negatively affect the liver. It is therefore very important to honor its maximum dosage of 4000 mg per day, or lower for people with risk factors. Taking paracetamol more than 10 days per month can be a risk for the liver. Monitoring liver values can help, but conclusive changes in your blood work might be delayed until initial damage has happened. A risk factor is alcohol consumption. It increases if the intake overlaps. To be safe, avoid taking paracetamol for 24 hours after alcohol consumption. NSAIDs have a much lower risk of affecting the liver negatively. Cardiovascular risks ASS is also prescribed as a blood thinner. All NSAIDs have this effect to some extent. However, for ASS, the blood thinning effect extends to more than a week after it has been discontinued. Surgeries should be avoided until that effect has subsided. It also increases the risk for hemorrhagic stroke. If you have migraine with aura, you might want to avoid ASS and Diclofenac. NSAIDs also have the risk to increase thrombosis. If you are in as risk group for that, you should consider avoiding Diclofenac. Paracetamol increases blood pressure which can be relevant if there are preexisting risks like already increased blood pressure. If you take ASS as a blood thinner. Take Aspirin at least 60 minutes before Metamizole. Otherwise, the blood thinning effect of the ASS might be suppressed. Effective application NSAIDs have a therapeutic ceiling for pain relief. You might not see an increased benefit beyond a dose of 200 mg or 400 mg for Ibuprofen. However, this ceiling does not apply for their anti-inflammatory effect, which might increase until 600 mg or 800 mg. Also, a higher dose than 400 mg can often be more effective to treat period pain. Higher doses can reduce the non-pain symptoms of migraine. Diclofenac is commonly used beyond its pain relief ceiling for rheumatoid arthritis. Take pain medication early and in a high enough dose. Several mechanisms can increase the benefit of pain medication. Knowing your effective dose and the early signs to take it is important. If you have early signs of a migraine attack, or you know that you are getting your period, it often makes sense to start the medication before the pain onset. Pain can have cascading effects in the body, and often there is a minimum amount of medication that you need to get a good effect, while a lower dose is almost ineffective. As mentioned before, you can combine an NSAIDs and Paracetamol. The effects of NSAIDs and Paracetamol can enhance each other, potentially reducing your required dose. In an emergency, it can be safe to combine both of their maximum dosage for a short time. With Ibuprofen and Paracetamol, you can alternate between them every three hours to soften the respective lows in the 6-hour cycle of each of them. Caffeine can support the pain relief. A cup of coffee or a double-espresso might be enough. Medication overuse headache Don’t use pain medication against headaches for more than 15 days a month. If you are using pain medication too often for headaches, you might develop a medication overuse headache (German: Medikamentenübergebrauchskopfschmerz). They can be reversed by taking a break from any pain medication. If you are using triptans (not further discussed here), the limit is 10 days instead of 15 days. While less likely, a medication overuse headache can also appear when treating a different pain than headaches. If you have more headache days than your painkillers allow treating, there are a lot of medications for migraine prophylaxis. Some, like Amitriptyline, can also be effective for a variety of other kinds headaches.

- Martín Abente Lahaye: [Call for Applicants] Flatseal at Igalia's Coding Experience 2026 (2026/03/03 14:42)
  Six years ago I released Flatseal. Since then, it has become an essential tool in the Flatpak ecosystem helping users understand and manage application permissions. But there's still a lot of work to do! I'm thrilled to share that my employer Igalia has selected Flatseal for its Coding Experience 2026 mentoring program. The Coding Experience is a grant program for people studying Information Technology or related fields. It doesn't matter if you're enrolled in a formal academic program or are self-taught. The goal is to provide you with real world professional experience by working closely with seasoned mentors. As a participant, you'll work with me to improve Flatseal, addressing long standing limitations and developing features needed for recent Flatpak releases. Possible areas of work include: Redesign and refactor Flatseal's permissions backend Support denying unassigned permissions Support reading system-level overrides Support USB devices lists permissions Support conditional permissions Support most commonly used portals This is a great opportunity to gain real-world experience, while contributing to open source and helping millions of users. Applications are open from February 23rd to April 3rd. Learn more and apply here!
- Matthew Garrett: To update blobs or not to update blobs (2026/03/03 03:09)
  A lot of hardware runs non-free software. Sometimes that non-free software is in ROM. Sometimes it's in flash. Sometimes it's not stored on the device at all, it's pushed into it at runtime by another piece of hardware or by the operating system. We typically refer to this software as "firmware" to differentiate it from the software run on the CPU after the OS has started1, but a lot of it (and, these days, probably most of it) is software written in C or some other systems programming language and targeting Arm or RISC-V or maybe MIPS and even sometimes x862. There's no real distinction between it and any other bit of software you run, except it's generally not run within the context of the OS3. Anyway. It's code. I'm going to simplify things here and stop using the words "software" or "firmware" and just say "code" instead, because that way we don't need to worry about semantics. A fundamental problem for free software enthusiasts is that almost all of the code we're talking about here is non-free. In some cases, it's cryptographically signed in a way that makes it difficult or impossible to replace it with free code. In some cases it's even encrypted, such that even examining the code is impossible. But because it's code, sometimes the vendor responsible for it will provide updates, and now you get to choose whether or not to apply those updates. I'm now going to present some things to consider. These are not in any particular order and are not intended to form any sort of argument in themselves, but are representative of the opinions you will get from various people and I would like you to read these, think about them, and come to your own set of opinions before I tell you what my opinion is. THINGS TO CONSIDER Does this blob do what it claims to do? Does it suddenly introduce functionality you don't want? Does it introduce security flaws? Does it introduce deliberate backdoors? Does it make your life better or worse? You're almost certainly being provided with a blob of compiled code, with no source code available. You can't just diff the source files, satisfy yourself that they're fine, and then install them. To be fair, even though you (as someone reading this) are probably more capable of doing that than the average human, you're likely not doing that even if you are capable because you're also likely installing kernel upgrades that contain vast quantities of code beyond your ability to understand4. We don't rely on our personal ability, we rely on the ability of those around us to do that validation, and we rely on an existing

(possibly transitive) trust relationship with those involved. You don’t know the people who created this blob, you likely don’t know people who do know the people who created this blob, these people probably don’t have an online presence that gives you more insight. Why should you trust them? If it’s in ROM and it turns out to be hostile then nobody can fix it ever The people creating these blobs largely work for the same company that built the hardware in the first place. When they built that hardware they could have backdoored it in any number of ways. And if the hardware has a built-in copy of the code it runs, why do you trust that that copy isn’t backdoored? Maybe it isn’t and updates would introduce a backdoor, but in that case if you buy new hardware that runs new code aren’t you putting yourself at the same risk? Designing hardware where you’re able to provide updated code and nobody else can is just a dick move5. We shouldn’t encourage vendors who do that. Humans are bad at writing code, and code running on ancilliary hardware is no exception. It contains bugs. These bugs are sometimes very bad. This paper describes a set of vulnerabilities identified in code running on SSDs that made it possible to bypass encryption secrets. The SSD vendors released updates that fixed these issues. If the code couldn’t be replaced then anyone relying on those security features would need to replace the hardware. Even if blobs are signed and can’t easily be replaced, the ones that aren’t encrypted can still be examined. The SSD vulnerabilities above were identifiable because researchers were able to reverse engineer the updates. It can be more annoying to audit binary code than source code, but it’s still possible. Vulnerabilities in code running on other hardware can still compromise the OS. If someone can compromise the code running on your wifi card then if you don’t have a strong IOMMU setup they’re going to be able to overwrite your running OS. Replacing one non-free blob with another non-free blob increases the total number of non-free blobs involved in the whole system, but doesn’t increase the number that are actually executing at any point in time. Ok we’re done with the things to consider. Please spend a few seconds thinking about what the tradeoffs are here and what your feelings are. Proceed when ready. I trust my CPU vendor. I don’t trust my CPU vendor because I want to, I trust my CPU vendor because I have no choice. I don’t think it’s likely that my CPU vendor has designed a CPU that identifies when I’m generating cryptographic keys and biases the RNG output so my keys are significantly weaker than they look, but it’s not literally impossible. I generate keys on it anyway, because what choice do I have? At some point I will buy a new laptop because Electron will no longer fit in 32GB of RAM and I will have to make the same affirmation of trust, because the alternative is that I just don’t have a computer. And in any case, I will be communicating with other people who generated their keys on CPUs I have no control over, and I will also be relying on them to be trustworthy. If I refuse to trust my CPU then I don’t get to computer, and if I don’t get to computer then I will be sad. I suspect I’m not alone here. Why would I install a code update on my CPU when my CPU’s job is to run my code in the first place? Because it turns out that CPUs are complicated and messy and they have their own bugs, and those bugs may be functional (for example, some performance counter functionality was broken on Sandybridge at release, and was then fixed with a microcode blob update) and if you update it your hardware works better. Or it might be that you’re running a CPU with speculative execution bugs and there’s a microcode update that provides a mitigation for that even if your CPU is slower when you enable it, but at least now you can run virtual machines without code in those virtual machines being able to reach outside the hypervisor boundary and extract secrets from other contexts. When it’s put that way, why would I not install the update? And the straightforward answer is that theoretically it could include new code that doesn’t act in my interests, either deliberately or not. And, yes, this is theoretically possible. Of course, if you don’t trust your CPU vendor, why are you buying CPUs from them, but well maybe they’ve been corrupted (in which case don’t buy any new CPUs from them either) or maybe they’ve just introduced a new vulnerability by accident, and also you’re in a position to determine whether the alleged security improvements matter to you at all. Do you care about speculative execution attacks if all software running on your system is trustworthy? Probably not! Do you need to update a blob that fixes something you don’t care about and which might introduce some sort of vulnerability? Seems like no! But there's a difference between a recommendation for a fully informed device owner who has a full understanding of threats, and a recommendation for an average user who just wants their computer to work and to not be ransomwared. A code update on a wifi card may introduce a backdoor, or it may fix the ability for someone to compromise your machine with a hostile access point. Most people are just not going to be in a position to figure out which is more likely, and there's no single answer that's correct for everyone. What we do know is that where vulnerabilities in this sort of code have been discovered, updates have tended to fix them - but nobody has flagged such an update as a real-world vector for system compromise. My personal opinion? You should make your own mind up, but also you shouldn't impose that choice on others, because your threat model is not necessarily their threat model. Code updates are a reasonable default, but they shouldn't be unilaterally imposed, and nor should they be blocked outright. And the best way to shift the balance of power away from vendors who insist on distributing non-free blobs is to demonstrate the benefits gained from them being free - a vendor who ships free code on their system enables their customers to improve their code and enable new functionality and make their hardware more attractive. It's impossible to say with absolute certainty that your security will be improved by installing code blobs. It's also impossible to say with absolute certainty that it won't. So far evidence tends to support the idea that most updates that claim to fix security issues do, and there's not a lot of evidence to support the idea that updates add new backdoors. Overall I'd say that providing the updates is likely the right default for most users - and that that should never be strongly enforced, because people should be allowed to define their own security model, and whatever set of threats I'm worried about, someone else may have a good reason to focus on different ones. Code that runs on the CPU before the OS is still usually described as firmware - UEFI is firmware even though it's executing on the CPU, which should give a strong indication that the difference between "firmware" and "software" is largely arbitrary ↩ And, obviously 8051 ↩ Because UEFI makes everything more complicated, UEFI makes this more complicated. Triggering a UEFI runtime service involves your OS jumping into firmware code at runtime, in the same context as the OS kernel. Sometimes this will trigger a jump into System Management Mode, but other times it won't, and it's just your kernel executing code that got dumped into RAM when your system booted. ↩ I don't understand most of the diff between one kernel version and the next, and I don't have time to read all of it either. ↩ There's a bunch of reasons to do this, the most reasonable of which is probably not wanting customers to replace the code and break their hardware and deal with the support overhead of that, but not being able to replace code running on hardware I own is always going to be an affront to me. ↩

- Mathias Bonn: Mahjongg: Second Year in Review (2026/03/02 19:41)
  Another year of work on Mahjongg is over. This was a pretty good year, with smaller improvements from several contributors. Let's take a look at what's new in Mahjongg 49.x. Game Session Restoration Thanks to contributions by François Godin, Mahjongg now remembers the previous game in progress before quitting. On startup, you have the option to resume the game or restart it. New Pause Screen Pausing a game used to only blank out the tiles and dim them. Since games restored on startup are paused, the lack of information was confusing. A new pause screen has since been added, with prominent buttons to resume/restart or quit. Thanks to Jeff Fortin for raising this issue! A new Escape keyboard shortcut for pausing the game has also been added, and the game now pauses automatically when opening menus and dialogs. New Game Rules Dialog Help documentation for Mahjongg has existed for a long time, but it always seemed less than ideal to open and read through when you just want to get started. Keeping the documentation up-to-date and translated was also difficult. A new Game Rules dialog has replaced it, giving a quick overview of what the game is about. Accessibility Improvements Tiles without a free long edge now shake when clicked, to indicate that they are not selectable. Tiles are also slightly dimmer in dark mode now, and follow the high contrast setting of the operating system. When

attempting to change the layout while a game is in progress, a confirmation dialog about ending the current game is shown. https://blogs.gnome.org/mathias/files/2026/03/mahjongg-tile-shake.webm Fixes and Modernizations Various improvements to the codebase have been made, and tests were added for the game algorithm and layout loading. Performance issues with larger numbers of entries in the Scores dialog were fixed, as well as an issue focusing the username entry at times when saving a score. Some small rendering issues related to fractional scaling were also addressed. Mahjongg used to load its tile assets using GdkPixbuf, but since that’s being phased out, it’s now using Rsvg directly instead. The upcoming GTK 4.22 release is introducing a new internal SVG renderer, GtkSvg, which we will hopefully start using in the near future. GNOME Circle Membership After a few rounds of reviews from Gregor Niehl and Tobias Bernard, Mahjongg was accepted into GNOME Circle. Mahjongg now has a page on apps.gnome.org, instructions for contributing and testing on welcome.gnome.org, as well as a new app icon by Tobias. Future Improvements The following items are next on the roadmap: Port the Scores dialog to the one provided by libgnome-games-support Use GtkSvg instead of Rsvg for rendering tile assets Look into adding support for keyboard navigation (and possibly gamepad support) Download Mahjongg The latest version of Mahjongg is available on Flathub. That’s all for now!

- Jussi Pakkanen: Discovering a new class of primes fur the fun of it (2026/02/26 22:31)
There are a lot of prime classes, such as left truncating primes, twin primes, mersenne primes, palindromic primes, emirp primes and so on. The Wikipedia page on primes lists many more. Recently I got to thinking (as one is wont to do) how difficult would it be to come up with a brand new one. The only reliable way to know is to try it yourself.The basic loopThe method I used was fairly straightforward:Download a list of the first one million primesLook at itTry to come up with a patternCheck if numbers from your pattern show up on OEISFind out they are notRejoiceCheck again more rigorouslyRealize they are in fact there in a slightly different formGo to 2Eventually I managed to come up with a prime category that is not in OEIS. Python code that generates them can be found in this repo. It may have bugs (I discovered several in the course of writing this post). The data below has not been independently validated.Faro primesIn magic terminology, a Faro shuffle is one that cuts a deck of cards in half and then interleaves the results. It is also known as a perfect shuffle. There are two different types of Faro shuffle, an in shuffle and an out shuffle. They have the peculiar property that if you keep repeating the same operation, eventually the deck returns to the original order.A prime p is a Faro prime if all numbers obtained by applying Faro shuffles (either in or out shuffles, but only one type) to its decimal representation are also prime. A Faro prime can be an Faro in prime, a Faro out prime or both. As an example, 19 is a Faro in prime, because a single in shuffle returns it to its original form. It is not an Faro out prime, because out shuffling it produces 91, which is not a prime (91 = 7*13).The testing for this was not rigorous, but at least OEIS does not recognize it.StatisticsI only used primes with an even number of digits. For odd number of digits you'd first need to decide how in and out shuffles should work. This is left as an exercise to the reader.Within the first one milllion primes, there are 7492 in primes, 775 out primes and 38 that are both in and out primes.The numbers with one or two digits are not particularly interesting. The first "actual" Faro in prime is 1103. It can be in shuffled once yielding 1013.For the first out shuffle you need to go to 111533, which shuffles to 513131 and 153113.The first prime longer than 2 digits that qualifies for both a Faro in and out prime is 151673. Its in shuffle primes are 165713, 176153 and 117563. The corresponding out shuffle primes are 151673, 617531 and 563117.Within the first one million primes the largest in shuffle prime is 15484627, the largest out shuffle prime is 11911111 and the largest in and out prime is 987793.Further questionsAs is typical in maths, finding out something immediately raises more questions. For example:Why are there so many fewer out primes than in primes?How would this look for primes with odd number of digits in them?Is it possible to build primes by a mixture of in and out shuffles?Most of the primes do not complete a "full shuffle", that is, they repeat faster than a deck of fully unique playing cards would. For any number n can you

find a Faro prime that requires that many shuffles or is there an upper limit for the number of shuffles?

- Matthias Clasen: An update on SVG in GTK (2026/02/25 12:07)
  In my last post on this topic, I explained the history of SVG in GTK, and how I tricked myself into working on an SVG renderer in 2025. Now we are in 2026, and on the verge of the GTK 4.22 release. A good time to review how far we've come. Testsuites While working on this over the last year, I was constantly looking for good tests to check my renderer against. Eventually, I found the resvg testsuite, which has broad coverage and is refreshingly easy to work with. In my unscientific self-evaluation, GtkSvg passes 1250 of the 1616 tests in this testsuite now, which puts GTK one tier below where the web browsers are. It would be nice to catch up with them, but that will require closing some gaps in our rendering infrastructure to support more complex filters. The resvg testsuite only covers static SVG. Another testsuite that I've used a lot is the much older SVG 1.1 testsuite, which covers SVG animation. GtkSvg passes most of these tests as well, which I am happy about — animation was one of my motivations when going into this work. https://blogs.gnome.org/gtk/files/2026/02/Screencast-From-2026-02-24-20-45-33.webm Benchmarks But doing a perfect job of rendering complex SVG doesn't do us much good if it slows GTK applications down too much. Recently, we've started to look at the performance implications of SVG rendering. We have a 'scrolling wall of icons' benchmark in our gtk4-demo app, which naturally is good place to test the performance impact of icon rendering changes. When switching it over to GtkSvg, it initially dropped from 60fps to around 40 on my laptop. We've since done some optimizations and regained most of the lost fps. The performance impact on typical applications will be much smaller, since they don't usually present walls of icons in their UI. Stressing our rendering infrastructure with some more demanding content was another motivation when I started to work on SVG, so I think I can declare success here. Content Creators The new SVG renderer needs new SVGs to take advantage of the new capabilities. Thankfully, Jakub Steiner has been hard at work to update many of the symbolic icons in GNOME. Others are exploring what we can do with the animation capabilities of the new renderer. Expect these things to start showing up in apps over the next cycle. Future work Feature-wise, GtkSvg is more than good enough for all our icon rendering needs, so making it cover more obscure SVG features may not be big priority in the short term. GtkSvg will be available in GTK 4.22, but we will not use it for every SVG icon yet — we still have a much simpler symbolic icon parser which is used for icons that are looked up by icon name from an icontheme. Switching over to using GtkSvg for everything is on the agenda for the next development cycle, after we've convinced ourselves that we can do this without adverse effects on performance or resource consumption of apps. Ongoing improvements of our rendering infrastructure will help ensure that that is the case. Where you can help One of the most useful contributions is feedback on what does or doesn't work, so please: try out GtkSvg, and tell us if you find SVGs that are rendered badly or with poor performance! Update: GtkSvg is an unsandboxed, in-process SVG parser written in C, so we don't recommend using it for untrusted content — it is meant for trusted content such as icons, logos and other application resources. If you want to load a random SVG of unknown providence, please use a proper image loading framework like glycin (but still, tell us if you find SVGs that crash GtkSvg). Of course, contributions to GtkSvg itself are more than welcome too. Here is a list of possible things to work on. If you are interested in working on an application, the simple icon editor that ships with GTK really needs to be moved to its own project and under separate maintainership. If that sounds appealing to you, please get in touch. If you would like to support the GNOME foundation, who's infrastructure and hosting GTK relies on, please donate.
- Sam Thursfield: Status update, 23rd February 2026 (2026/02/23 09:14)
  Its moments of change that remain striking in your memory when you look back. I feel like i'm in a long period of change, and if like me you participate in the tech industry and open source then you probably feel the same. It's going to be a wild time to look back on. As humans we're

naturally drawn to exciting new changes. Its not just the tech industry. The Spanish transport minister recently announced ambicious plans to run trains at record speeds of 350km/h. Then two tragic accidents happened, apparently due to careless infrastructure maintenance. Its easy (and valid) to criticise the situation. But I can sympathise too. You don't see many news reports saying "Infrastructure is being maintained really well at the moment and there haven't been any accidents for years". We all just take that shit for granted. This is a "middle aged man states obvious truths" post, so here's another one we forget in the software world: Automating work doesn't make it go away. Lets say you automate a 10 step release process which takes an hour to do manually. That's pretty great, now at release time you just push a button and wait. Maybe you can get on with some other work meanwhile — except you still need to check the automation finished and the release published correctly. What if step 5 fails? Now you have drop your other work again, push that out of your brain and try to remember how the release process worked, which will be hazy enough if you've stopped ever doing release manually. Sometimes I'll take an hour of manual work each month in preference to maintaining a complex, bespoke automation system. Over time we do build great tools and successfully automate bits of our jobs. Forty or fifty years ago, most computer programmers could write assembly code and do register allocation in their heads. I can't remember the last time I needed that skill. The C compiler does it for me. The work of CPU register allocation hasn't gone away, though. I've outsourced the cognitive load to researchers and compiler teams working at places like IBM / Red Hat, embecosm and Apple who maintain GCC and LLM. When I first got into computer programming, at the tail end of the "MOV AX, 10h; INT 13h" era, part of the fun was this idea you could have wild ideas and simply create yourself piece by piece, making your own tools, and pulling yourself up by your bootstraps. Look at this teenager who created his own 3D game engine! Look at this crazy dude who made an entire operating system! Now I'm gonna do something cool that will change the world, and then ideally retire. It took me the longest time to see that this "rock star" development model is all mythology. Just like actual rock stars, in fact. When a musician appears with stylish clothes and a bunch of great songs, the "origin story" is a carefully curated myth. The music world is a diverse community of artists, stylists, mentors, coaches, co-writers, producers, technicians, drivers, promotors, photographers, session musicians and social media experts, constantly trading our skills and ideas and collaborating to make them a reality. Nobody just walks out of their bedroom onto a stage and changes the world. But that doesn't make for a good press release does it ? The AI bubble is built on this same myth of the individual creator. I think LLMs are a transformative tool, and computer programming will never be the same; the first time you input some vaguely worded English prompt and get back a working unit test, you see a shining road ahead paved with automation, where you can finally turn ideas into products within days or weeks instead of having to chisel away at them painfully for years. But here's the reality: our monkey brains are still the same size, and you can't If your new automation is flaky, then you're going to spend as much time debugging and fixing things as you always did. Doing things the old way may take longer, but the limiting factor was never our typing speed, but our capacity to understand and communicate new ideas."The future belongs to idea guys who can just do things". No it doesnt mate, the past, present and future belongs to diverse groups of people whose skills and abilities complement each other and who have collectively agreed on some sort of common goal. But that idea doesn't sell very well. If and when we do land on genuinely transformative new tool — something like a C compiler, or hypertext — then I promise you, everyone's going to be on it in no time. How long did it take for ChatGPT to go from 0 to 1 billlion users wordwide? In all of this, I've had an intense few months in a new role at Codethink. It's been an intense winter too — by some measures Galicia is literally the wettest place on earth right now — so I guess it was a good time to learn new things. Since I rejoined back in 2021 I've nearly always been outsourced on different client projects. What I'm learning now is how the company's R&D division works.

- Federico Mena-Quintero: Librsvg got its first AI slop pull request (2026/02/21 01:13)

You all know that librsvg is developed in gitlab.gnome.org, not in GitHub. The README prominently says, "PLEASE DO NOT SEND PULL REQUESTS TO GITHUB". So, of course, today librsvg got its first AI slop pull request and later a second one, both in GitHub. Fortunately (?) they were closed by the same account that opened them, four minutes and one minute after opening them, respectively. I looked. There is compiled Python code (nope, that's how you get another xz attack). There are uncomfortably large Python scripts with jewels like subprocess.run("a single formatted string") (nope, learn to call commands correctly). There are two vast JSON files with "suggestions" for branches to make changes to the code, with jewels like: Suggestions to call standard library functions that do not even exist. The proposed code does not even use the nonexistent standard library function. Adding enum variants to SVG-specific constructs for things that are not in the SVG spec. Adding incorrect "safety checks". assert!(!c_string.is_null()) to be replaced by if c_string.is_null() { return ""; }. Fix a "floating-point overflow"... which is already handled correctly, and with a suggestion to use a function that does not exist. Adding a cache for something that does not need caching (without an eviction policy (so it is a memory leak)). Parallelizing the entire rendering process through a 4-line function. Of course this does not work. Adding two "missing" filters from the SVG spec (they are already implemented), and the implementation is todo!(). It's all like that. I stopped looking, and reported both PRs for spam.

- Adrian Vovk: GNOME OS Hackfest @ FOSDEM 2026 (2026/02/18 16:57)
For a few days leading up to FOSDEM 2026, the GNOME OS developers met for a GNOME OS hackfest. Here are some of the things we talked about! Stable The first big topic on our to-do list was GNOME OS stable. We started by defining the milestone: we can call GNOME OS “stable” when we settle on a configuration that we’re willing to support long-term. The most important blocker here is systemd-homed: we know that we want the stable release of GNOME OS to use systemd-homed, and we don’t want to have to support pre-homed GNOME OS installations forever. We discussed the possiblity of building a migration script to move people onto systemd-homed once it’s ready, but it’s simply too difficult and dangerous to deploy this in practice. We did, however, agree that we can already start promoting GNOME OS a bit more heavily, provided that we make very clear that this is an unstsable product for very early adopters, who would be willing to occasionally reinstall their system (or manually migrate it). We also discussed the importance of project documentation. GNOME OS’s documentation isn’t in a great state at the moment, and this makes it especially difficult to start contributing. BuildStream, which is GNOME OS’s build system, has a workflow that is unfamiliar to most people that may want to contribute. Despite its comprehensive documentation, there’s no easy “quick start” reference for the most common tasks and so it is ultimately a source of friction for potential contributors. This is especially unfourtunate given the current excitement around building next-gen “distroless” operating systems. Our user documentation is also pretty sparse. Finally, the little documentation we do have is spread across different places (markdown comitted to git, GitLab Wiki pages, the GNOME OS website, etc) and this makes it very difficult for people to find it. Fixing /etc Next we talked about the situation with /etc on GNOME OS. /etc has been a bit of an unsolved problem in the UAPI group’s model of immutability: ideally all default configuration can be loaded from /usr, and so /etc would remain entirely for overrides by the system administrator. Unfourtunately, this isn’t currently the case, so we must have some solution to keep track of both upstream defaults and local changes in /etc. So far, GNOME OS had a complicated set-up where parts of /usr would be symlinked into /etc. To change any of these files, the user would have to break the symlinks and replace them with normal files, potentially requiring copies of entire directories. This would then cause loads of issues, where the broken symlinks cause /etc to slowly drift away from the changing defaults in /usr. For years, we’ve known that the solution would be overlayfs. This kernel filesystem allows us to mount the OS’s defaults underneath a writable layer for administrator overrides. For various reasons, however, we’ve struggled to deploy this in practice. Modern systemd has native support for

this arrangement via systemd-confext, and we decided to just give it a try at the hackfest. A few hours later, Valentin had a merge request to transition us to the new scheme. We’ve now fully rolled this out, and so the issue is solved in the latest GNOME OS nightlies. FEX and Flatpak Next, we discussed integrating FEX with Flatpak so that we can run x86 apps on ARM64 devices. Abderrahim kicked off the topic by telling us about fexwrap, a script that grafts two different Flatpak runtimes together to successfully run apps via FEX. After studying this implementation, we discussed what proper upstream support might look like. Ultimately, we decided that the first step will be a new Flatpak runtime extension that bundles FEX, the required extra libraries, and the “thunks” (glue libraries that let x86 apps call into native ARM GPU drivers). From there, we’ll have to experiment and see what integrations Flatpak itself needs to make everything work seamlessly. Abderrahim has already started hacking on this upstream. Amutable The Amutable crew were in Brussels for FOSDEM, and a few of them stopped in to attend our hackfest. We had some very interesting conversations! From a GNOME OS perspective, we’re quite excited about the potential overlap between our work and theirs. We also used the opportunity to discuss GNOME OS, of course! For instance, we were able to resolve some kernel VFS blockers for GNOME OS delta updates and Flatpak v2. mkosi For a few years, we’ve been exploring ways to factor out GNOME OS’s image build scripts into a reusable component. This would make it trivial for other BuildStream-based projects to distribute themselves as UAPI.3 DDIs. It would also allow us to ship device-specific builds of GNOME OS, which are necessary to target mobile devices like the Fairphone 5. At Boiling the Ocean 7, we decided to try an alternative approach. What if we could drop our bespoke image build steps, and just use mkosi? There, we threw together a prototype and successfully booted to login. With the concept proven, I put together a better prototype in the intervening months. This prompted a discussion with Daan, the maintainer of mkosi, and we ultimately decided that mkosi should just have native BuildStream support upstream. At the hackfest, Daan put together a prototype for this native support. We were able to use his modified build of mkosi to build a freedesktop-sdk BuildStream image, package it up as a DDI, boot it in a virtual machine, set the machine up via systemd-firstboot, and log into a shell. Daan has since opened a pull request, and we’ll continue iterating on this approach in the coming months. Overall, this hackfest was extremely productive! I think it’s pretty likely that we’ll organize something like this again next year!

- Jonathan Blandford: Crosswords 0.3.17: Circle Bound (2026/02/16 07:30)
  It’s time for another Crosswords release. This is relatively soon after the last one, but I have an unofficial rule that Crosswords is released after three bloggable features. We’ve been productive and blown way past that bar in only a few months, so it’s time for an update. This round, we redid the game interface (for GNOME Circle) and added content to the editor. The editor also gained printing support, and we expanded support for Adwaita accent colors. In details: New Layout GNOME Crosswords’ new look — now using the accent color I applied for GNOME Circle a couple years ago, but it wasn’t until this past GUADEC that I was able to sit down together with Tobias to take a closer look at the game. We sketched out a proposed redesign, and I’ve been implementing it for the last four months. The result: a much cleaner look and workflow. I really like the way it has grown. Initial redesign Overall, I’m really happy with the way it looks and feels so far. The process has been relatively smooth (details), though it’s clear that the design team has limited resources to spend on these efforts. They need more help, and I hope that team can grow. Here’s how the game looks now: https://blogs.gnome.org/jrb/files/2026/02/main-1.webm I really could use help with the artwork for this project! Jakub made some sketches and I tried to convert them to svg, but have reached the limits of my inkscape skills. If you’re interested in helping and want to get involved in GNOME Design artwork, this could be a great place to start. Let me know! Indicator Hints Time for some crossword nerdery: Indicator Hints Dialog Main Screen One thing that characterizes cryptic crosswords is that its clues feature wordplay. A key part of the wordplay is called an “indicator hint”. These hints are a word — or words — that tell you to transform neighboring words into parts of the

solutions. These transformations could be things like rearranging the letters (anagrams) or reversing them. The example in the dialog screenshot below might give a better sense of how these work. There's a whole universe built around this. Indicator Hint Dialog with an example Good clues always use evocative indicator hints to entertain or mislead the solver. To help authors, I install a database of common indicator hints compiled by George Ho and show a random subset. His list also includes how frequently they're used, which can be used to make a clue harder or easier to solve. Indicator Hints Dialog with full list of indicators Templates and Settability I've always been a bit embarrassed about the New Puzzle dialog. The dialog should be simple enough: select a puzzle type, puzzle size, and maybe a preset grid template. Unfortunately, it historically had a few weird bugs and the template thumbnailing code was really slow. It could only render twenty or so templates before the startup time became unbearable. As a result, I only had a pitiful four or five templates per type of puzzle. When Toluwaleke rewrote the thumbnail rendering to be blazing fast over the summer, it became possible to give this section a closer look. The result: https://blogs.gnome.org/jrb/files/2026/02/greeter.webm Note: The rendering issues with the theme words dialog is GTK Bug #7400 The new dialog now has almost a thousand curated blank grids to pick from, sorted by how difficult they are to fill. In addition, I added initial support to add Theme Words to the puzzle. Setting theme words will also filter the templates to only show those that fit. Some cool technical details: The old dialog would load the ipuz files, convert them to svg, then render them to Pixbuf. That had both json + xml parse trees to navigate, plus a pixbuf transition. It was all inherently slow. I've thrown all that out. The new code takes advantage of the fact that crossword grids are effectively bitfields: at build time I convert each row in a grid template into a u32 with each bit representing a block. That means that each crossword grid can be stored as an array of these u32s. We use GResource and GVariant to load this file, so it's mmapped and effectively instant to parse. At this point, the limiting factor in adding additional blank templates is curation/generation. As part of this, I developed a concept called "settability" (documentation) to capture how easy or hard it is to fill in a grid. We use this to sort the grids, and to warn the user should they choose a harder grid. It's a heuristic, but it feels pretty good to me. You can see it in the video in the sort order of the grids. User Testing I had the good fortune to be able to sit with one of my coworkers and watch her use the editor. She's a much more accomplished setter than I, and publishes her crosswords in newspapers. Watching her use the tool was really helpful as she highlighted a lot of issues with the application (list). It was also great to validate a few of my big design decisions, notably splitting grid creation from clue writing. I've fixed most of the easy issues she found, but she confirmed something I suspected: The big missing feature for the editor is an overlay indicating tricky cells and dead ends (bug). Victor proposed a solution (link) for this over the summer. This is now the top priority for the next release. Thanks George for his fabulous database of indicator words Tobias for tremendous design work Jakub for artwork sketches and ideas Sophia for user feedback with the editor Federico for a lot of useful advice, CI fixes, and cleanups Vinson for build fixes and sanitation Nicole for some game papercut fixes Toluwaleke for printing advice and fixes Rosanna for text help and encouragement/advice Victor for cleaning up the docs Until next time!

- Cassidy James Blaede: How I Designed My Cantina Birthday Party (2026/02/14 00:00)
  Ever since my partner and I bought a house several years ago, I've wanted to throw a themed Star Wars party here. We've talked about doing a summer movie showing thing, we've talked about doing a Star Wars TV show marathon, and we've done a few birthday parties—but never the full-on themed party that I was dreaming up. Until this year! For some reason, a combination of rearranging some of our furniture, the state of my smart home, my enjoyment of Star Wars: Outlaws, and my newfound work/life balance meant that this was the year I finally committed to doing the party. Pitch For the past few years I've thrown a two-part birthday party: we start out at a nearby bar or restaurant, and then head to the house for more drinks and games. I like this format as it gives folks a natural "out" if they don't want to commit to the entire evening: they

can just join the beginning and then head out, or they can just meet up at our house. I was planning to do the same this year, but decided: let’s go all-in at the house so we have more time for more fun. I knew I wanted: Trivia! I organized a fun little Star Wars trivia game for my birthday last year and really enjoyed how nerdy my friends were with it, so this year I wanted to do something similar. My good friend Dagan volunteered to put together a fresh trivia game, which was incredible. Sabacc. The Star Wars equivalent to poker, featured heavily in the Star Wars: Outlaws game as well as in Star Wars: Rebels, Solo: A Star Wars Story, and the Disney Galactic Starcruiser (though it’s Kessel sabacc vs. traditional sabacc vs. Corellian spike vs. Coruscant shift respectively… but I digress). I got a Kessel sabacc set for Christmas and have wanted to play it with a group of friends ever since. Themed drinks. Revnog is mentioned in Star Wars media including Andor as some sort of liquor, and spotchka is featured in the New Republic era shows like The Mandalorian and The Book of Boba Fett. There isn’t really any detail as to what each tastes like, but I knew I wanted to make some batch cocktails inspired by these in-universe drinks. Immersive environment. This meant smart lights, music, and some other aesthetic touches. Luckily over the years I’ve upgraded my smart home to feature nearly all locally-controllable RGB smart bulbs and fixtures; while during the day they simply shift from warm white to daylight and back, it means I can do a lot with them for special occasions. I also have networked speakers throughout the house, and a 3D printer. About a month before the party, I got to work. Aesthetic For the party to feel immersive, I knew getting the aesthetic right was paramount. I also knew I wanted to send out themed invites to set the tone, so I had to start thinking about the whole thing early. Star Wars: Outlaws title screen Star Wars: Outlaws journal UI Since I’d been playing Star Wars: Outlaws, that was my immediate inspiration. I also follow the legendary Louie Mantia on Mastodon, and had bought some of his Star Wars fonts from The Crown Type Company, so I knew at least partially how I was going to get there. Initial invite graphic (address censored) For the invite, I went with a cyan-on-black color scheme. This is featured heavily in Star Wars: Outlaws but is also an iconic Star Wars look (“A long time ago…”, movie end credits, Clone Wars title cards, etc.). I chose the Spectre font as it’s very readable but also very Star Wars. To give it some more texture (and as an easter egg for the nerds), I used Womprat Aurebesh offset and dimmed behind the heading. The whole thing was a pretty quick design, but it did its job and set the tone. Website I spent a bit more time iterating on the website, and it’s a more familiar domain for me than more static designs like the invite was. I especially like how the offset Aurebesh turned out on the headings, as it feels very in-universe to me. I also played with a bit of texture on the website to give it that lo-fi/imperfect tech vibe that Star Wars so often embraces. For the longer-form body text, I wanted something even more readable than the more display-oriented fonts I’d used, so I turned to a good friend: Inter (also used on this site!). It doesn’t really look like Inter though… because I used almost every stylistic alternate that the font offers—explicitly to make it feel legible but also… kinda funky. I think it worked out well. Specifically, notice the lower-case “a”, “f”, “L”, “t”, and “u” shapes, plus the more rounded punctuation. Screenshot of my website I think more people should use subdomains for things like this! It’s become a meme at this point that people buy domains for projects they never get around to, but I always have to remind people: subdomains are free. Focus on making the thing, put it up on a subdomain, and then if you ever spin it out into its own successful thing, then you can buy a flashy bare domain for it! Since I already owned blaede.family where I host extended family wishlists, recipes, and a Mastodon server, I resisted the urge to purchase yet another domain and instead went with a subdomain. cantina.blaede.family doesn’t quite stay totally immersive, but it worked well enough—especially for a presumably short-lived project like this. Environment Once I had the invite nailed down, I started working on what the actual physical environment would look like. I watched the bar/cantina scenes from A New Hope and Attack of the Clones, scoured concept art, and of course played more Outlaws. The main thing I came away thinking about was lighting! Lighting The actual cantinas are often not all that otherworldly, but lighting plays a huge role; both in color and the overall dimness with a lot of (sometimes colorful) accent lighting. So, I got to work on setting up a lighting scene in Home Assistant. At first I was using the same color scheme everywhere, but I quickly found that distinct color schemes for different areas would feel more fun and interesting. Lounge area For the main lounge-type area, I went with dim orange lighting and just a couple of green accent lamps. This reminds me of Jabba's palace and Boba Fett, and just felt… right. It's sort of organic but would be a somewhat strange color scheme outside of Star Wars. It's also the first impression people will get when coming into the house, so I wanted it to feel the most recognizably Star Wars-y. Kitchen area Next, I focused on the kitchen, where people would gather for drinks and snacks. We have white under-cabinet lighting which I wanted to keep for function (it's nice to see what color your food actually is…), but I went with a bluish-purple (almost ultaviolet) and pink. Coruscant bar from Attack of the Clones While this is very different from a cantina on Tatooine, it reminded me of the Coruscant bar we see in Attack of the Clones as well as some of the environments in The Clone Wars and Outlaws. At one point I was going to attempt to make a glowing cocktail that would luminesce under black light—I ditched that, but the lighting stayed. Dining room sabacc table One of the more important areas was, of course, the sabacc table (the dining room), which is adjacent to the kitchen. I had to balance ensuring the cards and chips are visible with that dim, dingy, underworld vibe. I settled on actually adding a couple of warm white accent lights (3D printed!) for visibility, then using the ceiling fan lights as a sabacc round counter (with a Zigbee button as the dealer token). 3D printed accent light Lastly, I picked a few other colors for adjacent rooms: a more vivid purple for the bathroom, and red plus a rainbow LED strip for my office (where I set up split-screen Star Wars: Battlefront II on a PS2). Office area I was pretty happy with the lighting at this point, but then I re-watched the Mos Eisley scenes and noticed some fairly simple accent lights: plain warm white cylinders on the tables. I threw together a simple print for my 3D printer and added some battery-powered puck lights underneath: perfection. First test of my cylinder lights Music With my networked speakers, I knew I wanted some in-universe cantina music—but I also knew the cantina song would get real old, real fast. Since I'd been playing Outlaws as well as a fan-made Holocard Cantina sabacc app, I knew there was a decent amount of in-universe music out there; luckily it's actually all on YouTube Music. I made a looooong playlist including a bunch of that music plus some from Pyloon's Saloon in Jedi: Survivor, Oga's Cantina at Disney's Galaxy's Edge, and a select few tracks from other Star Wars media (Niamos!). Sabacc A big part of the party was sabacc; we ended up playing several games and really getting into it. To complement the cards and dice (from Hyperspace Props), I 3D printed chips and tokens that we used for the games. 3D printed sabacc tokens and chips We started out simple with just the basic rules and no tokens, but after a couple of games, we introduced some simple tokens to make the game more interesting. Playing sabacc I had a blast playing sabacc with my friends and by the end of the night we all agreed: we need to play this more frequently than just once a year for my birthday! Drinks I'm a fan of batch cocktails for parties, because it means less time tending a bar and more time enjoying company—plus it gives you a nice opportunity for a themed drink or two that you can prepare ahead of time. I decided to make two batch cocktails: green revnog and spotchka. Bottles of spotchka and revnog Revnog is shown a few times in Andor, but it's hard to tell what it looks like—one time it appears to be blue, but it's also lit by the bar itself. When it comes to taste, the StarWars.com Databank just says it "comes in a variety of flavors." However, one character mentions "green revnog" as being her favorite, so I decided to run with that so I could make something featuring objectively the best fruit in the galaxy: pear (if you know, you know). My take on green revnog After a lot of experimenting, I settled on a spiced pear gin drink that I think is a nice balance between sweet, spiced, and boozy. The simple batch recipe came out to: 4 parts gin, 1 part St. George's Spiced Pear Liqueur, 1 part pear juice, and 1 part lemon juice. It can be served directly on ice, or cut with sparkling water to tame it a bit. Spotchka doesn't get its own StarWars.com Databank entry, but is mentioned in a couple of entries about locations from an arc of The Mandalorian. All that can be gleaned is that it's apparently glowing and blue (Star Wars sure loves its blue drinks!), and made from "krill" which in Star Wars is shrimp-like. My take on spotchka I knew blue curaçao would be critical for a blue cocktail, and after a bit of asking around for inspiration, I decided coconut cream would give it a nice opacity and lightness. The obvious other ingredients for me, then, were rum and pineapple juice. I wanted it to taste a little more complex than just a Malibu pineapple, so I raided my liquor supply until I found my "secret" ingredient: grapefruit vodka. Just a tiny bit of that made it taste really unique and way more interesting! The final ratios for the batch are: 4 parts coconut rum, 2 parts white rum, 2 parts blue curaçao, 1 part grapefruit vodka, 2 parts pineapple juice, 1 part coconut cream. Similar to the revnog, it can be served directly on ice or cut with sparkling water for a less boozy drink. Summary Over all I had a blast hanging out, drinking cocktails, playing sabacc, and nerding out with my friends. I feel like the immersive-but-not-overbearing environment felt right; just one friend (the trivia master!) dressed up, which was perfect as I explicitly told everyone that costumes were not expected but left it open in case anyone wanted to dress up. The trivia, drinks, and sabacc all went over well, and a handful of us hung around until after 2 AM enjoying each other's company. That's a win in my book. :)

- Martin Pitt: Revisiting Google Cloud Performance for KVM-based CI (2026/02/13 00:00)
  Summary from 2022 Back then, I evaluated Google Cloud Platform for running Cockpit's integration tests. Nested virtualization on GCE was way too slow, crashy, and unreliable for our workload. Tests that ran in 35-45 minutes on bare metal (my laptop) took over 2 hours with 15 failures, timeouts, and crashes. The nested KVM simply wasn't performant enough. On today's Day of Learning, I gave this another shot, and was pleasantly surprised.
- Olav Vitters: GUADEC 2026 accommodation (2026/02/12 08:51)
  One of the things that I appreciate in a GUADEC (if available) is a common accommodation. Loads of attendees appreciated the shared accommodation in Vilanova i la Geltrú, Spain (GUADEC 2006). For GUADEC 2026 Deepesha announced one recommended accommodation, a student's residence. GUADEC 2026 is at the same place as GUADEC 2012, meaning: A Coruña, Spain. I didn't go to the 2012 one though I heard it also had a shared accommodation. For those wondering where to stay, suggest the recommended one.
- Asman Malika: Career Opportunities: What This Internship Is Teaching Me About the Future (2026/02/09 14:04)
  Before Outreachy, when I thought about career opportunities, I mostly thought about job openings, applications, and interviews. Opportunities felt like something you wait for, or hope to be selected for. This internship has changed how I see that completely. I'm learning that opportunities are often created through contribution, visibility, and community, not just applications. Opportunities Look Different in Open Source Working with GNOME has shown me that contributing to open source is not just about writing code, it's about building a public track record. Every merge request, every review cycle, every improvement becomes part of a visible body of work. Through my work on Papers: implementing manual signature features, fixing issues, contributing to Poppler codebase and now working on digital signatures, I'm not just completing tasks. I'm building real-world experience in a production codebase used by actual users. That kind of experience creates opportunities that don't always show up on job boards: Collaborating with experienced maintainers Learning large-project workflows Becoming known within a technical community Developing credibility through consistent contributions Skills That Expand My Career Options This internship is also expanding what I feel qualified to do.I'm gaining experience with: Building new features Large, existing codebases Code review and iteration cycles Debugging build failures and integration issues Writing clearer documentation and commit messages Communicating technical progress These are skills that apply across many roles, not just one job title. They open doors to remote collaboration, open-source roles, and product-focused engineering work. Career Is Bigger Than Employment One mindset shift for me is that career is no longer just about "getting hired." It's also about impact and direction. I now think more about: What kind of software I want to help build What communities I want to contribute to How accessible and user-

focused tools can be How I can support future newcomers the way my GNOME mentors supported me Open source makes career feel less like a ladder and more like a network. Creating Opportunities for Others Coming from a non-traditional path into tech, I’m especially aware of how powerful access and guidance can be. Programs like Outreachy don’t just create opportunities for individuals, they multiply opportunities through community. As I grow, I want to contribute not only through code, but also through sharing knowledge, documenting processes, and encouraging others who feel unsure about entering open source. Looking Ahead I don’t have every step mapped out yet. But I now have something better: direction and momentum. I want to continue contributing to open source, deepen my technical skills, and work on tools that people actually use. Outreachy and GNOME have shown me that opportunities often come from showing up consistently and contributing thoughtfully. That’s the path I plan to keep following.

- Christian Hergert: Mid-life transitions (2026/02/06 23:37)
The past few months have been heavy for many people in the United States, especially families navigating uncertainty about safety, stability, and belonging. My own mixed family has been working through some of those questions, and it has led us to make a significant change. Over the course of last year, my request to relocate to France while remaining in my role moved up and down the management chain at Red Hat for months without resolution, ultimately ending in a denial. That process significantly delayed our plans despite providing clear evidence of the risks involved to our family. At the beginning of this year, my wife and I moved forward by applying for long-stay visitor visas for France, a status that does not include work authorization. During our in-person visa appointment in Seattle, a shooting involving CBP occurred just a few parking spaces from where we normally park for medical outpatient visits back in Portland. It was covered by the news internationally and you may have read about it. Moments like that have a way of clarifying what matters and how urgently change can feel necessary. Our visas were approved quickly, which we’re grateful for. We’ll be spending the next year in France, where my wife has other Tibetan family. I’m looking forward to immersing myself in the language and culture and to taking that responsibility seriously. Learning French in mid-life will be humbling, but I’m ready to give it my full focus. This move also means a professional shift. For many years, I’ve dedicated a substantial portion of my time to maintaining and developing key components across the GNOME platform and its surrounding ecosystem. These projects are widely used, including in major Linux distributions and enterprise environments, and they depend on steady, ongoing care. For many years, I’ve been putting in more than forty hours each week maintaining and advancing this stack. That level of unpaid or ad-hoc effort isn’t something I can sustain, and my direct involvement going forward will be very limited. Given how widely this software is used in commercial and enterprise environments, long-term stewardship really needs to be backed by funded, dedicated work rather than spare-time contributions. If you or your organization depend on this software, now is a good time to get involved. Perhaps by contributing engineering time, supporting other maintainers, or helping fund long-term sustainability. The folliwing is a short list of important modules where I’m roughly the sole active maintainer: GtkSourceView – foundation for editors across the GTK eco-system Text Editor – GNOME’s core text editor Ptyxis – Default terminal on Fedora, Debian, Ubuntu, RHEL/CentOS/Alma/Rocky and others libspelling – Necessary bridge between GTK and enchant2 for spellcheck Sysprof – Whole-systems profiler integrating Linux perf, Mesa, GTK, Pango, GLib, WebKit, Mutter, and other statistics collectors Builder – GNOME’s flagship IDE template-glib – Templating and small language runtime for a scriptable GObject Introspection syntax jsonrpc-glib – Provides JSONRPC communication with language servers libpeas – Plugin library providing C/C++/Rust, Lua, Python, and JavaScript integration libdex – Futures, Fibers, and io_uring integration GOM – Data object binding between GObject and SQLite Manuals – Documentation reader for our development platform Foundry – Basically Builder as a command-line program and shared library, used by Manuals and a future Builder (hopefully) d-spy – Introspect D-Bus

connections libpanel – Provides IDE widgetry for complex GTK/libadwaita applications libmks – Qemu Mouse-Keyboard-Screen implementation with DMA-BUF integration for GTK There are, of course, many other modules I contribute to, but these are the ones most in need of attention. I'm committed to making the transition as smooth as possible and am happy to help onboard new contributors or teams who want to step up. My next chapter is about focusing on family and building stability in our lives.

- Lucas Baudin: Being a Mentor for Outreachy (2026/02/06 20:03)
  I first learned about Outreachy reading Planet GNOME 10 (or 15?) years ago. At the time, I did not know much about free software and I was puzzled by this initiative, as it mixed politics and software in a way I was not used to. Now I am a mentor for the December 2025 Outreachy cohort for Papers (aka GNOME Document Viewer), so I figured I would write a blog post to explain what Outreachy is and perpetuate the tradition! Furthermore, I thought it might be interesting to describe my experience as a mentor so far. What is Outreachy? Quoting the Outreachy website: Outreachy provides [paid] internships to anyone from any background who faces underrepresentation, systemic bias, or discrimination in the technical industry where they are living. These internships are paid and carried out in open-source projects. By way of anecdote, it was initially organized by the GNOME community around 2006-2009 to encourage women participation in GNOME and was progressively expanded to other projects later on. It was formally renamed Outreachy in 2015 and is now managed independently on GNOME, apart from its participation as an open-source project. Compared to the well-funded Summer of Code program by Google, Outreachy has a much more precarious financial situation, especially in recent years. With little surprise, the evolution of politics in the US and elsewhere over the last few years does not help. Therefore, most internships are nowadays funded directly by open-source projects (in our case the GNOME Foundation, you can donate and become a Friend of GNOME), and Outreachy still has to finance (at least) its staff (donations here). Outreachy as a Mentor So, I am glad that the GNOME Foundation was able to fund an Outreachy internship for the December 2025 cohort. As I am one of the Papers maintainers, I decided to volunteer to mentor an intern and came up with a project on document signatures. This was one of the first issues filled when Papers was forked from Evince, and I don't think I need to elaborate on how useful PDF signing is nowadays. Furthermore, Tobias had already made designs for this feature, so I knew that if we actually had an intern, we would precisely know what needed to be implemented1. Once the GNOME Internship Committee for Outreachy approved the project, the project was submitted on the Outreachy website, and applicants were invited to start making contributions to projects during the month of October so projects could then select interns (and interns could decide whether they wanted to work for three months in this community). Applicants were already selected by Outreachy (303 applications were approved out of 3461 applications received). We had several questions and contributions from around half a dozen applicants, and that was already an enriching experience for me. For instance, it was interesting to see how newcomers to Papers could be puzzled by our documentation. At this point, a crucial thing was labeling some issues as "Newcomers". It is much harder than what it looks (because sometimes things that seem simple actually aren't), and it is necessary to make sure that issues are not ambiguous, as applicants typically do not dare to ask questions (even, of course, when it is specified that questions are welcomed!). Communication is definitively one of the hardest things. In the end, I had to grade applicants (another hard thing to do), and the Internship Committee selected Malika Asman who accepted to participate as an intern! Malika wrote about her experience so far in several posts in her blog. 1 Outreachy internships do not have to be centered around programming; however, that is what I could offer guidance for.
- Matthias Clasen: GTK hackfest, 2026 edition (2026/02/06 10:43)
  As is by now a tradition, a few of the GTK developers got together in the days before FOSDEM to make plans and work on your favorite toolkit.

Code We released gdk-pixbuf 2.44.5 with glycin-based XPM and XBM loaders, rounding out the glycin transition. Note that the XPM/XBM support in will only appear in glycin 2.1. Another reminder is that gdk_pixbuf_new_from_xpm_data()was deprecated in gdk-pixbuf 2.44, and should not be used any more, as it does not allow for error handling in case the XPM loader is not available; if you still have XPM assets, please convert them to PNG, and use GResource to embed them into your application if you don't want to install them separately. We also released GTK 4.21.5, in time for the GNOME beta release. The highlights in this snapshot are still more SVG work (including support for SVG filters in CSS) and lots of GSK renderer refactoring. We decided to defer the session saving support, since early adopters found some problems with our APIs; once the main development branch opens for GTK 4.24, we will work on a new iteration and ask for more feedback. Discussions One topic that we talked about is unstable APIs, but no clear conclusion was reached. Keeping experimental APIs in the same shared object was seen as problematic (not just because of ABI checkers). Making a separate shared library (and a separate namespace, for bindings) might not be easy. Still on the topic of APIs, we decided that we want to bump our C runtime requirement to C11 in the next cycle, to take advantage of standard atomics, integer types and booleans. At the moment, C11 is a soft requirement through GLib. We also talked about GLib's autoptrs, and were saddened by the fact that we still can't use them without dropping MSVC. The defer proposal for C2y would not really work with how we use automatic cleanup for types, either, so we can't count on the C standard to save us. Mechanics We collected some ideas for improving project maintenance. One idea that came up was to look at automating issue tagging, so it is easier for people to pay closer attention to a subset of all open issues and MRs. Having more accurate labels on merge requests would allow people to get better notifications and avoid watching the whole project. We also talked about the state of GTK3 and agreed that we want to limit changes in this very mature code base to crash and build fixes: the chances of introducing regressions in code that has long since been frozen is too high. Accessibility On the accessibility side, we are somewhat worried about the state of AccessKit. The code upstream is maintained, but we haven't seen movement in the GTK implementation. We still default to the AT-SPI backend on Linux, but AccessKit is used on Windows and macOS (and possibly Android in the future); it would be nice to have consumers of the accessibility stack looking at the code and issues. On the AT-SPI side we are still missing proper feature negotiation in the protocol; interfaces are now versioned on D-Bus, but there's no mechanism to negotiate the supported set of roles or events between toolkits, compositors, and assistive technologies, which makes running newer applications on older OS versions harder. We discussed the problem of the ARIA specification being mostly "stringly" typed in the attributes values, and how it impacts our more strongly typed API (especially with bindings); we don't have a good generic solution, so we will have to figure out possible breaks or deprecations on a case by case basis. Finally, we talked about a request by the LibreOffice developers on providing a wrapper for the AT-SPI collection interface; this API is meant to be used as a way to sidestep the array-based design, and perform queries on the accessible objects tree. It can be used to speed up iterating through large and sparse trees, like documents or spreadsheets. It's also very AT-SPI specific, which makes it hard to write in a platform-neutral way. It should be possible to add it as a platform-specific API, like we did for GtkAtSpiSocket. Carlos is working on landing the pointer query API in Mutter, which would address the last remnant of X11 use inside Orca. Outlook Some of the plans and ideas that we discussed for the next cycle include: Bring back the deferred session saving Add some way for applications to support the AT-SPI collection interface Close some API gaps in GtkDropDown (8003 and 8004) Bring some general purpose APIs from libadwaita back to GTK Until next year,

- Cassidy James Blaede: ROOST at FOSDEM 2026 (2026/02/06 00:00)
  A few months ago I joined ROOST (Robust Open Online Safety Tools) to build our open source community that would be helping to create, distribute, and maintain common tools and building blocks for online trust and safety. One of the first events I wanted to make sure we attended

in order to build that community was of course FOSDEM, the massive annual gathering of open source folks in Brussels, Belgium. Luckily for us, the timing aligned nicely with the v1 release of our first major online safety tool, Osprey, as well as its adoption by Bluesky and the Matrix.org Foundation. I wrote and submitted a talk for the FOSDEM crowd and the decentralized communications track, which was accepted. Our COO Anne Bertucio and I flew out to Brussels to meet up with folks, make connections, and learn how our open source tools could best serve open protocols and platforms. Brunch with the Christchurch Call Foundation Saturday, ROOST co-hosted a brunch with the Christchurch Call Foundation where we invited folks to discuss the intersection of open source and online safety. The event was relatively small, but we engaged in meaningful conversations and came away with several recurring themes. Non-exhaustively, some areas attendees were interested in: novel classifiers for unique challenges like audio recordings and pixel art; how to ethically source and train classifiers; ways to work better together across platforms and protocols. Personally I enjoyed meeting folks from Mastodon, GitHub, ATproto, IFTAS, and more in person for the first time, and I look forward to continuing several conversations that were started over coffee and fruit. Talk Our Sunday morning talk “Stop Reinventing in Isolation” (which you can watch on YouTube or at fosdem.org) filled the room and was really well-received. View on YouTube Cassidy and Anne giving a talk. | Photos from @matrix@mastodon.matrix.org In it we tackled three major topics: a crash course on what is “trust and safety”; why the field needs an open source approach; and then a bit about Osprey, our self-hostable automated rules engine and investigation tool that started as an internal tool built at Discord. Q&A We had a few minutes for Q&A after the talk, and the folks in the room spurred some great discussions. If there’s something you’d like to ask that isn’t covered by the talk or this Q&A, feel free to start a discussion! Also note that this gets a bit nerdy; if you’re not interested in the specifics of deploying Osprey, feel free to skip ahead to the Stand section. When using Osprey with the decentralized Matrix protocol, would it be a policy server implementation? Yes, in the Matrix model that’s the natural place to handle it. Chat servers are designed to check with the policy server before sending room events to clients, so it’s precisely where you’d want to be able to run automated rules. The Matrix.org Foundation is actively investigating how exactly Osprey can be used with this setup, and already have it deployed in their staging environment for testing. Does it make sense to use Osprey for smaller platforms with fewer events than something like Matrix, Bluesky, or Discord? This one’s a bit harder to answer, because Osprey is often the sort of tool you don’t “need” until you suddenly and urgently do. That said, it is designed as an in-depth investigation tool, and if that’s not something needed on your platform yet due to the types and volume of events you handle, it could be overkill. You might be better off starting with a moderation/review dashboard like Coop, which we expect to be able to release as v0 in the coming weeks. As your platform scales, you could then explore bringing Osprey in as a complementary tool to handle more automation and deeper investigation. Does Osprey support account-level fraud detection? Osprey itself is pretty agnostic to the types of events and metadata it handles; it’s more like a piece of plumbing that helps you connect a firehose of events to one end, write rules and expose those events for investigation in the middle, and then connect outgoing actions on the other end. So while it’s been designed for trust and safety uses, we’ve heard interest from platforms using it in a fraud prevention context as well. What are the hosting requirements of Osprey, and what do deployments look like? While you can spin Osprey up on a laptop for testing and development, it can be a bit beefy. Osprey is made up of four main components: worker, UI, database, and Druid as the analytics database. The worker and UI have low resource requirements, your database (e.g. Postgres) could have moderate requirements, but then Druid is what will have the highest requirements. The requirements will also scale with your total throughput of events being processed, as well as the TTLs you keep in Druid. As for deployments, Discord, Bluesky, and the Matrix.org Foundation have each integrated Osprey into their Kubernetes setups as the components are fairly standard Docker images. Osprey also comes with an optional coordinator, an action distribution and load-balancing service that can aid with horizontal scaling. Stand This year we were unable to secure a stand (there were already nearly 100 stands in just 5 buildings!), but our friends at Matrix graciously hosted us for several hours at their stand near the decentralized communications track room so we could follow up with folks after our talk. We blew through our shiny sticker supply as well as our 3D printed ROOST keychains (which I printed myself at home!) in just one afternoon. We'll have to bring more to future FOSDEMs! When I handed people one of our hexagon stickers the reaction was usually some form of, "ooh, shiny!" but my favorite was when someone essentially said, "Oh, you all actually know open source!" That made me proud, at least. :) Interesting Talks Lastly, I always like to shout out interesting talks I attended or caught on video later so others can enjoy them on their own time. I recommend checking out: Community moderation in Matrix Open Source Security in Spite of AI

- Mathias Bonn: The Hobby Lives On (2026/01/28 04:17)
  Maintaining an open source project in your free time is incredibly rewarding. A large project full of interesting challenges, limited only by your time and willingness to learn. Years of work add up to something you've grown proud of. Who would've thought an old project on its last legs could turn into something beautiful? The focus is intense. So many people using the project, always new things to learn and improve. Days fly by when time allows for it. That impossible feature sitting in the backlog for years, finally done. That slow part of the application, much faster now. This flow state is pretty cool, might as well tackle a few more issues while it lasts. Then comes the day. The biggest release yet is out the door. More tasks remain on the list, but it's just too much. That release took so much effort, and the years are adding up. You can't keep going like this. You wonder, is this the beginning of the end? Will you finally burn out, like so many before you? A smaller project catches your eye. Perhaps it would be fun to work on something else again. Maybe it doesn't have to be as intense? Looks like this project uses a niche programming language. Is it finally time to learn another one? It's an unfamiliar project, but it's pretty fun. It tickles the right spots. All the previous knowledge helps. You work on the smaller project for a while. It goes well. That larger project you spent years on lingers. So much was accomplished. It's not done yet, but software is never done. The other day, someone mentioned this interesting feature they really wanted. Maybe it wouldn't hurt to look into it? It's been a while since the last feature release. Maybe the next one doesn't have to be as intense? It's pretty fun to work on other projects sometimes, too. The hobby lives on. It's what you love doing, after all.
- Asman Malika: Mid-Point Project Progress: What I've Learned So Far (2026/01/26 13:42)
  Dark mode: Manual Signature Implementation Light mode: When there is no added signature Reaching the midpoint of this project feels like a good moment to pause, not because the work is slowing down, but because I finally have enough context to see the bigger picture. At the start, everything felt new: the codebase, the community, the workflow, and even the way problems are framed in open source. Now, halfway through, things are starting to connect. Where I Started When I began working on Papers, my main focus was understanding the codebase and how contributions actually happen in a real open-source project. Reading unfamiliar code, following discussions, and figuring out where my work fit into the larger system was challenging. Early on, progress felt slow. Tasks that seemed small took longer than expected, mostly because I was learning how the project works, not just what to code. But that foundation has been critical. Photo: Build failure I encountered during development What I've Accomplished So Far At this midpoint, I'm much more comfortable navigating the codebase and understanding the project's architecture. I've worked on the manual signature feature and related fixes, which required carefully reading existing implementations, asking questions, and iterating based on feedback. I'm now working on the digital signature implementation, which is one of the most complext part of the project and builds directly on the foundation laid by the earlier work. Beyond the technical work, I've learned how collaboration really functions in open source: How to communicate progress clearly How to receive and apply feedback How to break down problems instead of

rushing to solutions These skills have been just as important as writing code. Challenges Along the Way One of the biggest challenges has been balancing confidence and humility, knowing when to try things independently and when to ask for help. I've also learned that progress in open source isn't always linear. Some days are spent coding, others reading, debugging, or revisiting decisions. Another challenge has been shifting my mindset from "just making it work" to thinking about maintainability, users, and future contributors. That shift takes time, but it's starting to stick. What's Changed Since the Beginning The biggest change is how I approach problems. I now think more about who will use the feature, who might read this code later, and how my changes fit into the overall project. Thinking about the audience, both users of Papers and fellow contributors, has influenced how I write code, documentation, and even this blog. I'm also more confident participating in discussions and expressing uncertainty when I don't fully understand something. That confidence comes from realizing that learning in public is part of the process. Looking Ahead The second half of this project feels more focused. With the groundwork laid, I can move faster and contribute more meaningfully. My goal is to continue improving the quality of my contributions, take on more complex tasks, and deepen my understanding of the project. Most importantly, I want to keep learning about open source, about collaboration, and about myself as a developer. Final Thoughts This midpoint has reminded me that growth isn't always visible day to day, but it becomes clear when you stop and reflect. I'm grateful for the support, feedback, and patience from GNOME community, especially my mentor Lucas Baudin. And I'm so excited to see how the rest of the project unfolds.

- Christian Schaller: Can AI help 'fix' the patent system? (2026/01/21 18:35)
  So one thing I think anyone involved with software development for the last decades can see is the problem of "forest of bogus patents". I have recently been trying to use AI to look at patents in various ways. So one idea I had was "could AI help improve the quality of patents and free us from obvious ones?" Lets start with the justification for patents existing at all. The most common argument for the patent system I hear is this one : "Patents require public disclosure of inventions in exchange for protection. Without patents, inventors would keep innovations as trade secrets, slowing overall technological progress.". This reasoning is something that makes sense to me, but it is also screamingly obvious to me that for it to hold true you need to ensure the patents granted are genuinely inventions that otherwise would stay hidden as trade secrets. If you allow patents on things that are obvious to someone skilled in the art, you are not enhancing technological progress, you are hampering it because the next person along will be blocking from doing it. So based on this justification the question then becomes does for example the US Patents Office do a good job filtering out obvious patents? And I believe the answer is "No they don't". Having been working in the space of software for many decades now it is very clear to me that the patent office do very little to avoid patents getting approved for obvious things. And there are many reasons for why that happens I think. First of all if you are an expert in a field you would likely not be working as a case worker in the patent office, no disrespect for the people working there, but for instance if you are a really good software engineer there are much better paying options to be found elsewhere, same goes for medical researchers for example. So we have people who have limited understanding of the field of the patent trying to evaluate if the patent is obvious or not. Secondly the patent office is not really incentivized to deny patents as they are a self funded entity and their 'success' and revenue is based on how many patent applications are submitted. So the lower the bar of approval, the more patents applications get submitted and thus the patent office makes more money and is more "successful". This is a problem, but maybe if we fix the first issue I mention it is maybe possible to reduce the impact of this second issue. So what can AI do to help here? It can help patent reviewers at the patent office with evaluating patents for obviousness due to having been trained on these fields. If we required every patent applied for to be run through a AI engine for review then the patent office reviewer would have to justify why he or she

believes something is non-obvious after the AI has looked at it and concluded it is not. My assumption would be that they would very rarely do so. To test out this I decided to ask Gemini to review a H264 video patent that is still valid and which has been used in litigation before, to see if it could do a decent job of reviewing the patent application and judge if this is a true invention or not. Here is what Gemini came up with. US Patent 7,916,782 (Samsung Electronics) is one of the most famous "battleground" patents in the history of smartphone litigation. It has been asserted against Apple, Huawei, and others. It covers a specific part of the H.264 standard known as CAVLC (Context-Adaptive Variable Length Coding). The Patent: US 7,916,782 : Method and apparatus for video coding and decoding The Mechanism: Efficiently coding the count of zero-valued pixels in a video block ("TotalZeros"). The Problem In video compression, you divide the image into small 4×4 pixel blocks. After processing (transform and quantization), most of the 16 pixels in the block end up being zero. You only have a few non-zero numbers left. Goal: You need to tell the decoder how many zeros are in the block so it can reconstruct the empty space. Constraint: You already told the decoder how many non-zero coefficients exist (let's say you have 5 non-zeros). The "Invention" The patent claims a method where the encoder selects a specific lookup table (VLC table) to encode the "TotalZeros" value, and—crucially—this selection is based on the number of non-zero coefficients (TotalCoeff) that were just processed. If TotalCoeff is 1, use Table A (allows for many zeros).If TotalCoeff is 10, use Table B (allows for fewer zeros). The "Skilled Engineer" Test Imagine you are a video coding engineer in 2002. You are tasked with compressing the "TotalZeros" value. Here is the logical deduction chain you would inevitably follow: The Constraint Check (Basic Math) Engineer's Thought: "I am working with a 4×4 block. That's 16 pixels total." Engineer's Thought: "I just transmitted the number of non-zero pixels. Let's say there are 5 non-zeros. Deduction: "Therefore, the maximum number of zeros possible is 16−5=11." The Optimization (Common Sense) Engineer's Thought: "I need a Huffman table (lookup table) to compress the TotalZeros value." Engineer's Thought: "Should I use a generic table that goes from 0 to 16?" Deduction: "No, that's wasteful. I know for a fact the value cannot be 12, 13, 14, 15, or 16. If I use a table that includes those impossible numbers, I'm wasting bits on shorter codes for values that will never happen." The Solution (The "Invention") Engineer's Thought: "I should use a smaller, custom table that only contains values 0 through 11." Engineer's Thought: "Actually, I should just make a different table for every possible number of non-zeros. If I have 15 non-zeros, the only possible zero count is 1. I don't even need a table for that; it's implicit!" The Verdict: Valid or Broken? Engineer's View (Obvious): This is not an invention; it is constraint propagation. If you have a container of capacity 16, and you fill it with 5 rocks, it is "obvious" that the remaining space for air is 11. Creating a table sized to "11" isn't a creative spark; it is the mathematical consequence of the block size. To an engineer, failing to do this would be incompetence, not a design choice. Patent Office View (Novel): The USPTO granted this because the patent didn't just claim "optimizing tables." It claimed the specific structure of the solution: Check TotalCoeff.If TotalCoeff is 1, select Table_1. If TotalCoeff is 2, select Table_2. …and so on. Because no prior paper had explicitly written down "Use these specific tables in this specific order for H.264," it was deemed novel. Conclusion US 7,916,782 is a prime example of the "broken" system because it effectively patents arithmetic efficiency. By granting a monopoly on the logic "Use a small table when you have small space," the patent office allowed Samsung to assert ownership over the only logical way to implement the H.264 standard efficiently. An engineer trying to solve this problem without seeing the patent would essentially be forced to "re-invent" it, because the math leaves no other optimal path. Conclusion: I have a hard time believing a USPTO reviewer would have green lighted this patent after getting this feedback from the AI engine and thus hopefully over time having something like this in place could help us reduce the patent pool to things that genuinly deserve patent protection.

- Sebastian Wick: Best Practices for Ownership in GLib (2026/01/21 15:31)

For all the rightful criticisms that C gets, GLib does manage to alleviate at least some of it. If we can’t use a better language, we should at least make use of all the tools we have in C with GLib. This post looks at the topic of ownership, and also how it applies to libdex fibers. Ownership In normal C usage, it is often not obvious at all if an object that gets returned from a function (either as a real return value or as an out-parameter) is owned by the caller or the callee: MyThing *thing = my_thing_new (); If thing is owned by the caller, then the caller also has to release the object thing. If it is owned by the callee, then the lifetime of the object thing has to be checked against its usage. At this point, the documentation is usually being consulted with the hope that the developer of my_thing_new documented it somehow. With gobject-introspection, this documentation is standardized and you can usually read one of these: The caller of the function takes ownership of the data, and is responsible for freeing it. The returned data is owned by the instance. If thing is owned by the caller, the caller now has to release the object or transfer ownership to another place. In normal C usage, both of those are hard issues. For releasing the object, one of two techniques are usually employed: single exit MyThing *thing = my_thing_new (); gboolean c; c = my_thing_a (thing); if (c) c = my_thing_b (thing); if (c) my_thing_c (thing); my_thing_release (thing); /* release thing */ goto cleanup MyThing *thing = my_thing_new (); if (!my_thing_a (thing)) goto out; if (!my_thing_b (thing)) goto out; my_thing_c (thing); out: my_thing_release (thing); /* release thing */ Ownership Transfer GLib provides automatic cleanup helpers (g_auto, g_autoptr, g_autofd, g_autolist). A macro associates the function to release the object with the type of the object (e.g. G_DEFINE_AUTOPTR_CLEANUP_FUNC). If they are being used, the single exit and goto cleanup approaches become unnecessary: g_autoptr(MyThing) thing = my_thing_new (); if (!my_thing_a (thing)) return; if (!my_thing_b (thing)) return; my_thing_c (thing); The nice side effect of using automatic cleanup is that for a reader of the code, the g_auto helpers become a definite mark that the variable they are applied on own the object! If we have a function which takes ownership over an object passed in (i.e. the called function will eventually release the resource itself) then in normal C usage this is indistinguishable from a function call which does not take ownership: MyThing *thing = my_thing_new (); my_thing_finish_thing (thing); If my_thing_finish_thing takes ownership, then the code is correct, otherwise it leaks the object thing. On the other hand, if automatic cleanup is used, there is only one correct way to handle either case. A function call which does not take ownership is just a normal function call and the variable thing is not modified, so it keeps ownership: g_autoptr(MyThing) thing = my_thing_new (); my_thing_finish_thing (thing); A function call which takes ownership on the other hand has to unset the variable thing to remove ownership from the variable and ensure the cleanup function is not called. This is done by “stealing” the object from the variable: g_autoptr(MyThing) thing = my_thing_new (); my_thing_finish_thing (g_steal_pointer (&thing)); By using g_steal_pointer and friends, the ownership transfer becomes obvious in the code, just like ownership of an object by a variable becomes obvious with g_autoptr. Ownership Annotations Now you could argue that the g_autoptr and g_steal_pointer combination without any conditional early exit is functionally exactly the same as the example with the normal C usage, and you would be right. We also need more code and it adds a tiny bit of runtime overhead. I would still argue that it helps readers of the code immensely which makes it an acceptable trade-off in almost all situations. As long as you haven’t profiled and determined the overhead to be problematic, you should always use g_auto and g_steal! The way I like to look at g_auto and g_steal is that it is not only a mechanism to release objects and unset variables, but also annotations about the ownership and ownership transfers. Scoping One pattern that is still somewhat pronounced in older code using GLib, is the declaration of all variables at the top of a function: static void foobar (void) { MyThing *thing = NULL; size_t i; for (i = 0; i < len; i++) { g_clear_pointer (&thing); thing = my_thing_new (i); my_thing_bar (thing); } } We can still avoid mixing declarations and code, but we don’t have to do it at the granularity of a function, but of natural scopes: static void foobar (void) { for (size_t i = 0; i < len; i++) { g_autoptr(MyThing) thing = NULL; thing = my_thing_new (i); my_thing_bar (thing); } } Similarly, we can introduce our own scopes which can be used to limit how long variables, and thus objects are alive: static void foobar (void) { g_autoptr(MyOtherThing) other = NULL; { /* we only need `thing` to get `other` */ g_autoptr(MyThing) thing = NULL; thing = my_thing_new (); other = my_thing_bar (thing); } my_other_thing_bar (other); } Fibers When somewhat complex asynchronous patterns are required in a piece of GLib software, it becomes extremely advantageous to use libdex and the system of fibers it provides. They allow writing what looks like synchronous code, which suspends on await points: g_autoptr(MyThing) thing = NULL; thing = dex_await_object (my_thing_new_future (), NULL); If this piece of code doesn't make much sense to you, I suggest reading the libdex Additional Documentation. Unfortunately the await points can also be a bit of a pitfall: the call to dex_await is semantically like calling g_main_loop_run on the thread default main context. If you use an object which is not owned across an await point, the lifetime of that object becomes critical. Often the lifetime is bound to another object which you might not control in that particular function. In that case, the pointer can point to an already released object when dex_await returns: static DexFuture * foobar (gpointer user_data) { /* foo is owned by the context, so we do not use an autoptr */ MyFoo *foo = context_get_foo (); g_autoptr(MyOtherThing) other = NULL; g_autoptr(MyThing) thing = NULL; thing = my_thing_new (); /* side effect of running g_main_loop_run */ other = dex_await_object (my_thing_bar (thing, foo), NULL); if (!other) return dex_future_new_false (); /* foo here is not owned, and depending on the lifetime * (context might recreate foo in some circumstances), * foo might point to an already released object */ dex_await (my_other_thing_foo_bar (other, foo), NULL); return dex_future_new_true (); } If we assume that context_get_foo returns a different object when the main loop runs, the code above will not work. The fix is simple: own the objects that are being used across await points, or re-acquire an object. The correct choice depends on what semantic is required. We can also combine this with improved scoping to only keep the objects alive for as long as required. Unnecessarily keeping objects alive across await points can keep resource usage high and might have unintended consequences. static DexFuture * foobar (gpointer user_data) { /* we now own foo */ g_autoptr(MyFoo) foo = g_object_ref (context_get_foo ()); g_autoptr(MyOtherThing) other = NULL; { g_autoptr(MyThing) thing = NULL; thing = my_thing_new (); /* side effect of running g_main_loop_run */ other = dex_await_object (my_thing_bar (thing, foo), NULL); if (!other) return dex_future_new_false (); } /* we own foo, so this always points to a valid object */ dex_await (my_other_thing_bar (other, foo), NULL); return dex_future_new_true (); } static DexFuture * foobar (gpointer user_data) { /* we now own foo */ g_autoptr(MyOtherThing) other = NULL; { /* We do not own foo, but we only use it before an * await point. * The scope ensures it is not being used afterwards. */ MyFoo *foo = context_get_foo (); g_autoptr(MyThing) thing = NULL; thing = my_thing_new (); /* side effect of running g_main_loop_run */ other = dex_await_object (my_thing_bar (thing, foo), NULL); if (!other) return dex_future_new_false (); } { MyFoo *foo = context_get_foo (); dex_await (my_other_thing_bar (other, foo), NULL); } return dex_future_new_true (); } One of the scenarios where re-acquiring an object is necessary, are worker fibers which operate continuously, until the object gets disposed. Now, if this fiber owns the object (i.e. holds a reference to the object), it will never get disposed because the fiber would only finish when the reference it holds gets released, which doesn't happen because it holds a reference. The naive code also suspiciously doesn't have any exit condition. static DexFuture * foobar (gpointer user_data) { g_autoptr(MyThing) self = g_object_ref (MY_THING (user_data)); for (;;) { g_autoptr(GBytes) bytes = NULL; bytes = dex_await_boxed (my_other_thing_bar (other, foo), NULL); my_thing_write_bytes (self, bytes); } } So instead of owning the object, we need a way to re-acquire it. A weak-ref is perfect for this. static DexFuture * foobar (gpointer user_data) { /* g_weak_ref_init in the caller somewhere */ GWeakRef *self_wr = user_data; for (;;) { g_autoptr(GBytes) bytes = NULL; bytes = dex_await_boxed (my_other_thing_bar (other, foo), NULL); { g_autoptr(MyThing) self = g_weak_ref_get (&self_wr); if (!self) return dex_future_new_true (); my_thing_write_bytes (self, bytes); } } } Conclusion Always use g_auto/g_steal helpers to mark ownership and ownership transfers (exceptions do apply) Use scopes to limit the lifetime of objects In fibers, always own objects you need across await points, or re-acquire them

- Ignacy Kuchciński: Digital Wellbeing Contract: Conclusion (2026/01/20 03:00)
  A lot of progress has been made since my last Digital Wellbeing update two months ago. That post covered the initial screen time limits feature, which was implemented in the Parental Controls app, Settings and GNOME Shell. There’s a screen recording in the post, created with the help of a custom GNOME OS image, in case you’re interested. Finishing Screen Time Limits After implementing the major framework for the rest of the code in GNOME Shell, we added the mechanism in the lock screen to prevent children from unlocking when the screen time limit is up. Parents are now also able to extend the session limit temporarily, so that the child can use the computer until the rest of the day. Parental Controls Shield Screen time limits can be set as either a daily limit or a bedtime. With the work that has recently landed, when the screen time limit has been exceeded, the session locks and the authentication action is hidden on the lock screen. Instead, a message is displayed explaining that the current session is limited and the child cannot login. An “Ignore” button is presented to allow the parents to temporarily lift the restrictions when needed. Parental Controls shield on the lock screen, preventing the children from unlocking Extending Screen Time Clicking the “Ignore” button prompts the user for authentication from a user with administrative privileges. This allows parents to temporarily lift the screen time limit, so that the children may log in as normal until the rest of the day. Authentication dialog allowing the parents to temporarily override the Screen Time restrictions Showcase Continuing the screen cast of the Shell functionality from the previous update, I’ve recorded the parental controls shield together, and showed the extending screen time functionality: https://blogs.gnome.org/ignapk/files/2026/01/parental-controls-shield.webm GNOME OS Image You can also try the feature out for yourself, with the very same GNOME OS live image I’ve used in the recording, that you can either run in GNOME Boxes, or try on your hardware if you know what you’re doing Conclusion Now that the full Screen Time Limits functionality has been merged in GNOME Shell, this concludes my part in the Digital Wellbeing Contract. Here’s the summary of the work: We’ve redesigned the Parental Controls app and updated it to use modern GNOME technologies New features was added, such as Screen Time monitoring and setting limits: daily limit and bedtime schedule GNOME Settings gained Parental Controls integration, to helpfully inform the user about the existence of the limits We introduced the screen time limits in GNOME Shell, locking childrens’ sessions once they reach their limit. Children are then prevented from unlocking until the next day, unless parents extend their screen time In the initial plan, we also covered web filtering, and the foundation of the feature has been introduced as well. However, integrating the functionality in the Parental Controls application has been postponed to a future endeavour. I’d like to thank GNOME Foundation for giving me this opportunity, and Endless for sponsoring the work. Also kudos to my colleagues, Philip Withnall and Sam Hewitt, it’s been great to work with you and I’ve learned a lot (like the importance of wearing Christmas sweaters in work meetings!), and to Florian Müllner, Matthijs Velsink and Felipe Borges for very helpful reviews. I also want to thank Allan Day for organizing the work hours and meetings, and helping with my blog posts as well Until next project!
- Sriram Ramkrishna: GNOME OS Hackfest During FOSDEM week (2026/01/17 23:17)
  For those of you who are attending FOSDEM, we’re doing a GNOME OS hackfest and invite those of you who might be interested on our experiments on concepts as the ‘anti-distro’, eg an OS with no distro packaging that integrates GNOME desktop patterns directly. The hackfest is from January 28th – January 29th. If you’re interested, feel free to respond on the comments. I don’t have an exact location yet. We’ll likely have some kind of BigBlueButton set up so if you’re not available to come in-person you can join us remotely. Agenda and attendees are linked here here. There is likely a limited capacity so acceptance will be “first come, first served”. See you there!
- Gedit Technology: gedit 49.0 released (2026/01/16 10:00)

gedit 49.0 has been released! Here are the highlights since version 48.0 which dates back from September 2024. (Some sections are a bit technical). File loading and saving enhancements A lot of work went into this area. It's mostly under-the-scene changes where there was a lot of dusty code. It's not entirely finished, but there are already user-visible enhancements: Loading a big file is now much faster. gedit now refuses to load very big files, with a configurable limit (more details). Improved preferences There is now a "Reset All..." button in the Preferences dialog. And it is now possible to configure the default language used by the spell-checker. Python plugins removal Initially due to an external factor, plugins implemented in Python were no longer supported. During some time a previous version of gedit was packaged in Flathub in a way that still enabled Python plugins, but it is no longer the case. Even though the problem is fixable, having some plugins in Python meant to deal with a multi-language project, which is much harder to maintain for a single individual. So for now it's preferable to keep only the C language. So the bad news is that Python plugins support has not been re-enabled in this version, not even for third-party plugins. More details. Summary of changes for plugins The following plugins have been removed: Bracket Completion Character Map Color Picker Embedded Terminal Join/Split Lines Multi Edit Session Saver Only Python plugins have been removed, the C plugins have been kept. The Code Comment plugin which was written in Python has been rewritten in C, so it has not disappeared. And it is planned and desired to bring back some of the removed plugins. Summary of other news Lots of code refactorings have been achieved in the gedit core and in libgedit-gtksourceview. A better support for Windows. Web presence at gedit-text-editor.org: new domain name and several iterations on the design. A half-dozen Gedit Development Guidelines documents have been written. Wrapping-up statistics for 2025 The total number of commits in gedit and gedit-related git repositories in 2025 is: 884. More precisely: 138 enter-tex 310 gedit 21 gedit-plugins 10 gspell 4 libgedit-amtk 41 libgedit-gfls 290 libgedit-gtksourceview 70 libgedit-tepl It counts all contributions, translation updates included. The list contains two apps, gedit and Enter TeX. The rest are shared libraries (re-usable code available to create other text editors). If you do a comparison with the numbers for 2024, you'll see that there are fewer commits, the only module with more commits is libgedit-gtksourceview. But 2025 was a good year nevertheless! For future versions: superset of the subset With Python plugins removed, the new gedit version is a subset of the previous version, when comparing approximately the list of features. In the future, we plan to have a superset of the subset. That is, to bring in new features and try hard to not remove any more functionality. In fact, we have reached a point where we are no longer interested to remove any more features from gedit. So the good news is that gedit will normally be incrementally improved from now on without major regressions. We really hope there won't be any new bad surprises due to external factors! Side note: this "superset of the subset" resembles the evolution of C++, but in the reverse order. Modern C++ will be a subset of the superset to have a language in practice (but not in theory) as safe as Rust (it works with compiler flags to disable the unsafe parts). Onward to 2026 Since some plugins have been removed, this makes gedit a less advanced text editor. It has become a little less suitable for heavy programming workloads, but for that there are lots of alternatives. Instead, gedit could become a text editor of choice for newcomers in the computing science field (students and self-learners). It can be a great tool for markup languages too. It can be your daily companion for quite a while, until your needs evolve for something more complete at your workplace. Or it can be that you prefer its simplicity and its not-going-in-the-way default setup, plus the fact that it launches quickly. In short, there are a lot of reasons to still love gedit ❤ ! If you have any feedback, even for a small thing, I would like to hear from you :) ! The best places are on GNOME Discourse, or GitLab for more actionable tasks (see the Getting in Touch section).

- Ignacio Casal Quinteiro: Mecalin (2026/01/15 19:13)
  Many years ago when I was a kid, I took typing lessons where they introduced me to a program called Mecawin. With it, I learned how to type,

and it became a program I always appreciated not because it was fancy, but because it showed step by step how to work with a keyboard. Now the circle of life is coming back: my kid will turn 10 this year. So I started searching for a good typing tutor for Linux. I installed and tried all of them, but didn't like any. I also tried a couple of applications on macOS, some were okish, but they didn't work properly with Spanish keyboards. At this point, I decided to build something myself. Initially, I hacked out keypunch, which is a very nice application, but I didn't like the UI I came up with by modifying it. So in the end, I decided to write my own. Or better yet, let Kiro write an application for me. Mecalin is meant to be a simple application. The main purpose is teaching people how to type, and the Lessons view is what I'll be focusing on most during development. Since I don't have much time these days for new projects. I decided to take this opportunity to use Kiro to do most of the development for me. And to be honest, it did a pretty good job. Sure, there are things that could be better, but I definitely wouldn't have finished it in this short time otherwise. So if you are interested, give it a try, go to flathub and install it: https://flathub.org/apps/io.github.nacho.mecalin In this application, you'll have several lessons that guide you step by step through the different rows of the keyboard, showing you what to type and how to type it. This is an example of the lesson view. You also have games. The falling keys game: keys fall from top to bottom, and if one reaches the bottom of the window, you lose. This game can clearly be improved, and if anybody wants to enhance it, feel free to send a PR. The scrolling lanes game: you have 4 rows where text moves from right to left. You need to type the words before they reach the leftmost side of the window, otherwise you lose. For those who want to support your language, there are two JSON files you'll need to add: The keyboard layout: https://github.com/nacho/mecalin/tree/main/data/keyboard_layouts The lessons: https://github.com/nacho/mecalin/tree/main/data/lessons Note that the Spanish lesson is the source of truth; the English one is just a translation done by Kiro. If you have any questions, feel free to contact me.

- Flathub: What's new in Vorarbeiter (2026/01/14 00:00)
  It is almost a year since the switch to Vorarbeiter for building and publishing apps. We've made several improvements since then, and it's time to brag about them. RunsOn In the initial announcement, I mentioned we were using RunsOn, a just-in-time runner provisioning system, to build large apps such as Chromium. Since then, we have fully switched to RunsOn for all builds. Free GitHub runners available to open source projects are heavily overloaded and there are limits on how many concurrent builds can run at a time. With RunsOn, we can request an arbitrary number of threads, memory and disk space, for less than if we were to use paid GitHub runners. We also rely more on spot instances, which are even cheaper than the usual on demand machines. The downside is that jobs sometimes get interrupted. To avoid spending too much time on retry ping-pong, builds retried with the special bot, retry command use the on-demand instances from the get-go. The same catch applies to large builds, which are unlikely to finish in time before spot instances are reclaimed. The cost breakdown since May 2025 is as follows: Once again, we are not actually paying for anything thanks to the AWS credits for open source projects program. Thank you RunsOn team and AWS for making this possible! Caching Vorarbeiter now supports caching downloads and ccache files between builds. Everything is an OCI image if you are feeling brave enough, and so we are storing the per-app cache with ORAS in GitHub Container Registry. This is especially useful for cosmetic rebuilds and minor version bumps, where most of the source code remains the same. Your mileage may vary for anything more complex. End-of-life without rebuilding One of the Buildbot limitations was that it was difficult to retrofit pull requests marking apps as end-of-life without rebuilding them. Flat-manager itself exposes an API call for this since 2019 but we could not really use it, as apps had to be in a buildable state only to deprecate them. Vorarbeiter will now detect that a PR modifies only the end-of-life keys in the flathub.json file, skip test and regular builds, and directly use the flat-manager API to republish the app with the EOL flag set post-merge. Web UI GitHub's UI isn't really built for a centralized repository building other repositories. My love-hate relationship with Buildbot made me want to have a similar dashboard for Vorarbeiter. The

new web UI uses PicoCSS and HTMX to provide a tidy table of recent builds. It is unlikely to be particularly interesting to end users, but kinkshaming is not nice, okay? I like to know what's being built and now you can too here. Reproducible builds We have started testing binary reproducibility of x86_64 builds targetting the stable repository. This is possible thanks to flathub-repro-checker, a tool doing the necessary legwork to recreate the build environment and compare the result of the rebuild with what is published on Flathub. While these tests have been running for a while now, we have recently restarted them from scratch after enabling S3 storage for diffoscope artifacts. The current status is on the reproducible builds page. Failures are not currently acted on. When we collect more results, we may start to surface them to app maintainers for investigation. We also don't test direct uploads at the moment.

- Arun Raghavan: Accessibility Update: Enabling Mono Audio (2026/01/13 00:09)
  If you maintain a Linux audio settings component, we now have a way to globally enable/disable mono audio for users who do not want stereo separation of their audio (for example, due to hearing loss in one ear). Read on for the details on how to do this. Background Most systems support stereo audio via their default speaker output or 3.5mm analog connector. These devices are exposed as stereo devices to applications, and applications typically render stereo content to these devices. Visual media use stereo for directional cues, and music is usually produced using stereo effects to separate instruments, or provide a specific experience. It is not uncommon for modern systems to provide a “mono audio” option that allows users to have all stereo content mixed together and played to both output channels. The most common scenario is hearing loss in one ear. PulseAudio and PipeWire have supported forcing mono audio on the system via configuration files for a while now. However, this is not easy to expose via user interfaces, and unfortunately remains a power-user feature. Implementation Recently, Julian Bouzas implemented a WirePlumber setting to force all hardware audio outputs (MR 721 and 769). This lets the system run in stereo mode, but configures the audioadapter around the device node to mix down the final audio to mono. This can be enabled using the WirePlumber settings via API, or using the command line with: wpctl settings node.features.audio.mono true The WirePlumber settings API allows you to query the current value as well as clear the setting and restoring to the default state. I have also added (MR 2646 and 2655) a mechanism to set this using the PulseAudio API (via the messaging system). Assuming you are using pipewire-pulse, PipeWire’s PulseAudio emulation daemon, you can use pa_context_send_message_to_object() or the command line: pactl send-message /core pipewire-pulse:force-mono-output true This API allows for a few things: Query existence of the feature: when an empty message body is sent, if a null value is returned, feature is not supported Query current value: when an empty message body is sent, the current value (true or false) is returned if the feature is supported Setting a value: the requested setting (true or false) can be sent as the message body Clearing the current value: sending a message body of null clears the current setting and restores the default Looking ahead This feature will become available in the next release of PipeWire (both 1.4.10 and 1.6.0). I will be adding a toggle in Pavucontrol to expose this, and I hope that GNOME, KDE and other desktop environments will be able to pick this up before long. Hit me up if you have any questions!
- Zoey Ahmed: Welcome To The Coven! (2026/01/12 23:18)
  Introduction § Welcome to the long-awaited rewrite of my personal blog! It’s been 2 years since I touched the source code for my original website, and unfortunately in that time it’s fallen into decay, the source code sitting untouched for some time for a multitude of reasons. One of the main reasons for undertaking a re-write is I have changed a lot in the two years since I first started having my own blog. I have gained 2 years of experience and knowledge in fields like accessibility and web development, I became a regular contributor to the GNOME ecosystem, especially in the last half of 2025, I picked up playing music for myself and with friends in late 2024. I am now (thankfully) out as a transgender

woman to everyone in my life, and can use my website as a proper portfolio, rather then just as a nice home page for my friends to whom I was out the closet to. I began University in 2024 and gained a lot of web design experience in my second year, creating 2 (pretty nice) new websites in a short period for my group. In short, my previous website did not really reflect me or my passions anymore, and it sat untouched as the changes in my life added up. Another reason I undertook a rewrite was due to the frankly piss-poor architecture of my original website. My original website was all hand-written HTML and CSS! After it expanded a little, I tried to port what I had done with handwritten HTML/CSS to Zola, a static site generator. A static site generator, for those unfamiliar with the term, is a tool that takes markdown files, and some template and configuration files, and compiles them all into a set of static websites. In short, cutting down on the boilerplate and repeated code I would need to type every-time I made a new blog or subpage on my blog. I undertook the port to Zola in an attempt to make it easier to add new content to my blog, but it resulted in my website not taking full capability of the advantages of using a static site generator. I also disliked some parts about Zola, compared to other options like Jekyll and (the static site generator I eventually used in the rewrite) Hugo. On May 8th, 2025 I started rewriting my website, after creating a few designs in Penpot and getting feedback for their design by my close friends. This first attempt got about 80% to completion, but sat as I ran into a couple issues with making my website, and was overall unhappy with how some of the elements in my original draft of the rewrite came to fruition. One example was my portfolio: My old portfolio for Upscaler. It contains an image with 2 Upscaler windows, the image comparison mode in the left window, and the queue in the right window, with a description underneath. A pink border around it surrounds the image and description, with the project name and tags above the border I did not like the style of surrounding everything in large borders, the every portfolio item alternating between pink/purple was incredibly hard to do, and do well. I also didn’t take full advantage of things like subgrids in CSS, to allow me to make elements that were the full width of the page, while keeping the rest of the content dead centre. I also had trouble with making my page mobile responsive. I had a lot of new ideas for my blog, but never had time to get round to any of them, because I had to spend most my development time squashing bugs as I refactored large chunks of my website as my knowledge on Hugo and web design rapidly grew. I eventually let the rewrite rot for a few months, all while my original website was actually taken down for indefinite maintenance by my original hosting organization. On Janurary 8th, 2026, exactly 7 months after the rewrite was started, I picked up it up again, starting more or less from scratch, but resuing some components and most of the content from the first rewrite. I was armed with all the knowledge from my university group project’s websites, and inspired by my fellow GNOME contributors websites, including but not limited to: The Evil Skeleton Jeff Fortin Tam Georges Stavracas Tobias Bernard Laura Kramolis In just a couple of days, I managed to create something I was much more proud of. This can be seen within my portfolio page, for example: A screenshot of the top of portfolio page, with the laptop running GNOME and the section for Cartridges. I also managed to add many features and improvements I did not manage to first time around (all done with HTML/CSS, no Javascript!) such as a proper mobile menu, with animated drop downs and an animation playing when the button is clicked, a list of icons for my smaller GNOME contributions, instead of having an entire item dedicated to each, wasting vertical space, an adaptive friends of the site grid, and a cute little graphical of GNOME on a laptop at the top of my portfolio in the same style as Tobias Bernard’s and GNOME’s front page, screenshots switching from light/dark mode in the portfolio based on the users OS preferences and more. Overall, I am very proud of not only the results of my second rewriter, but how I managed to complete it in less than a week. I am happy to finally have a permanent place to call my own again, and share my GNOME development and thoughts in a place thats more collected and less ephemeral than something like my Fediverse account or (god forbid) a Bluesky or X account. Still, I have more work to do on my website front, like a proper light mode as pointed out by The Evil Skeleton, and to clean up my templates and 675 line long CSS file! For now, welcome to the re-introduction of my small area of the Internet, and prepare for yet another development blog by a GNOME developer.

- Engagement Team: GNOME ASIA 2025-Event Report (2026/01/09 11:35)
GNOME ASIA 2025 took place in Tokyo, Japan, from 13–14 December 2025, bringing together the GNOME community for the featured annual GNOME conference in Asia. The event was held in a hybrid format, welcoming both in-person and online speakers and attendees from across the world. GNOME ASIA 2025 was co-hosted with the LibreOffice Asia Conference community event, creating a shared space for collaboration and discussion between open-source communities. Photo by Tetsuji Koyama, licensed under CC BY 4.0 About GNOME.Asia Summit The GNOME.Asia Summit focuses primarily on the GNOME desktop while also covering applications and platform development tools. It brings together users, developers, foundation leaders, governments, and businesses in Asia to discuss current technologies and future developments within the GNOME ecosystem. The event featured 25 speakers in total, delivering 17 full talks and 8 lightning talks across the two days. Speakers joined both on-site and remotely. Photo by Tetsuji Koyama, licensed under CC BY 4.0 Around 100 participants attended in person in Tokyo, contributing to engaging discussions and community interaction. Session recordings were published on the GNOME Asia YouTube channel, where they have received 1,154 total views, extending the reach of the event beyond the conference dates. With strong in-person attendance, active online participation, and collaboration with the LibreOffice Asia community, GNOME ASIA 2025 once again demonstrated the importance of regional gatherings in strengthening the GNOME ecosystem and open-source collaboration in Asia. Photo by Tetsuji Koyama, licensed under CC BY 4.0
- Daiki Ueno: GNOME.Asia Summit 2025 (2026/01/07 08:36)
Last month, I attended the GNOME.Asia Summit 2025 held at the IIJ office in Tokyo. This was my fourth time attending the summit, following previous events in Taipei (2010), Beijing (2015), and Delhi (2016). As I live near Tokyo, this year’s conference was a unique experience for me: an opportunity to welcome the international GNOME community to my home city rather than traveling abroad. Reconnecting with the community after several years provided a helpful perspective on how our ecosystem has evolved. Addressing the post-quantum transition During the summit, I delivered a keynote address regarding post-quantum cryptography (PQC) and desktop. The core of my presentation focused on the “Harvest Now, Decrypt Later” (HNDL) type of threats, where encrypted data is collected today with the intent of decrypting it once quantum computing matures. The talk was followed by the history and the current status of PQC support in crypto libraries including OpenSSL, GnuTLS, and NSS, and concluded with the next steps recommended for the users and developers. It is important to recognize that classical public key cryptography, which is vulnerable to quantum attacks, is integrated into nearly every aspect of the modern desktop: from secure web browsing and apps using libsoup (Maps, Weather, etc.) to the underlying verification of system updates. Given that major government timelines (such as NIST and the NSA’s CNSA 2.0) are pushing for a full migration to quantum-resistant algorithms between 2027 and 2035, the GNU/Linux desktop should prioritize “crypto-agility” to remain secure in the coming decade. From discussion to implementation: Crypto Usage Analyzer One of the tools I discussed during my talk was crypto-auditing, a project designed to help developers identify and update the legacy cryptography usage. At the time of the summit, the tool was limited to a command-line interface, which I noted was a barrier to wider adoption. Inspired by the energy of the summit, I spent part of the recent holiday break developing a GUI for crypto-auditing. By utilizing AI-assisted development tools, I was able to rapidly prototype an application, which I call “Crypto Usage Analyzer”, that makes the auditing data more accessible. Conclusion The summit in Tokyo had a relatively small audience, which resulted in a cozy and professional atmosphere. This smaller scale proved beneficial for technical exchange, as it allowed for focused discussions on desktop-related topics than is often possible at larger conferences. Attending GNOME.Asia

2025 was a reminder of the steady work required to keep the desktop secure and relevant. I appreciate the efforts of the organizing committee in bringing the summit to Tokyo, and I look forward to continuing my work on making security libraries and tools more accessible for our users and developers.
- Christian Hergert: pgsql-glib (2026/01/02 20:54)
  Much like the s3-glib library I put together recently, I had another itch to scratch. What would it look like to have a PostgreSQL driver that used futures and fibers with libdex? This was something I wondered about more than a decade ago when writing the libmongoc network driver for 10gen (later MongoDB). pgsql-glib is such a library which I made to wrap the venerable libpq PostgreSQL state-machine library. It does operations on fibers and awaits FD I/O to make something that feels synchronous even though it is not. It also allows for something more “RAII-like” using g_autoptr() which interacts very nicely with fibers. API Documentation can be found here.