Gnome Planet

- Felipe Borges: One Project Selected for the December 2025 Outreachy Cohort with GNOME! (2025/12/03 11:03)
  We are happy to announce that the GNOME Foundation is sponsoring an Outreachy project for the December 2025 Outreachy cohort. Outreachy provides internships to people subject to systemic bias and impacted by underrepresentation in the tech industry where they are living. Let's welcome Malika Asman! Malika will be working with Lucas Baudin on improving document signing in Papers, our document viewer. The new contributor will soon get their blogs added to Planet GNOME making it easy for the GNOME community to get to know them and the projects that they will be working on. We would like to also thank our mentor, Lucas for supporting Outreachy and helping new contributors enter our project. If you have any questions, feel free to reply to this Discourse topic or message us privately at soc-admins@gnome.org.
- Cassidy James Blaede: Looking back on GNOME in 2025—and looking forward to 2026 (2025/12/02 00:00)
  This past year has been an exceptional one for GNOME. The project released two excellent releases on schedule with GNOME 48 in March and GNOME 49 in September. Contributors have been relentless in delivering a set of new and improved default apps, constant performance improvements across the board benefitting everyone (but especially lower-specced hardware), a better experience on high end hardware like HiDPI and HDR displays, refined design and refreshed typography, all new digital wellbeing features and parental controls improvements, improved accessibility support across the entire platform, and much more. Just take a look back through This Week in GNOME where contributors provided updates on development every single week of 2025 so far. (And a huge thank you to Felix, who puts This Week in GNOME together!) All of these improvements were delivered for free to users of GNOME across distributions—and even beyond users of GNOME itself via GNOME apps running on any desktop thanks to Flatpak and distribution via Flathub. Earlier this year the GNOME Foundation also relaunched Friends of GNOME where you can set up a small recurring donation to help fund initiatives including: infrastructure freely provided to Core, Circle, and World projects services for GNOME Foundation members like blog hosting, chat, and video conferencing development of Flathub community travel sponsorship While I'm proud of what GNOME has accomplished in 2025 and that the GNOME Foundation is operating sustainably, I'm personally even more excited to look ahead to what I hope the Foundation will be able to achieve in the coming year. Let's Reach 1,500 Friends of GNOME The newly-formed fundraising committee kicked off their efforts by announcing a simple goal to close out 2025: let's reach 1,500 Friends of GNOME! If we can reach this goal by the end of this year, it will help GNOME deliver even more in 2026; for example, by enabling the Foundation to sponsor more community travel for hackfests and conferences, and potentially even sponsoring specific, targeted development work. But GNOME needs your help! How You Can Help First, if you're not already a Friend of GNOME, please consider setting up a small recurring donation at donate.gnome.org. Every little bit helps, and donating less but consistently is super valuable to not only keep the lights on at the GNOME Foundation, but to enable explicit budgeting for and delivering on more interesting initiatives that directly support the community and the development of GNOME itself. Become a Friend of GNOME If you're already a Friend of GNOME (or not able to commit to that at the moment—no hard feelings!), please consider sharing this message far and wide! I consistently hear that not only do so many users of GNOME not know that it's a nonprofit, but they don't know that the GNOME Foundation relies on individual donations—and that users can help out, too! Please share this post to your circles—especially outside of usual contributor spaces—to let them know the cool things GNOME does and that GNOME could use their help to be able to do even more in the coming year. Lastly, if you represent an organization that relies on GNOME or is invested in its continued success, please consider a corporate sponsorship. While this sponsorship comes with no strings attached, it's a really powerful way to

show that your organization supports Free and Open Source software—and puts their money where their mouth is. Sponsor GNOME Thank You! Thank you again to all of the dedicated contributors to GNOME making everyone's computing experience that much better. As we close out 2025, I'm excited by the prospect of the GNOME Foundation being able to not just be sustainable, but—with your help—to take an even more active role in supporting our community and the development of GNOME. And of course, thank you to all 700+ current Friends of GNOME; your gracious support has helped GNOME achieve everything in 2025 while ensuring the sustainability of the Foundation going forward. Let's see if we can close out the year with 1,500 Friends helping GNOME do even more!

- Christian Hergert: Status Week 48 (2025/12/01 21:53)
  This week was Thanksgiving holiday in the US and I spent the entire week quite sick with something that was not Covid nor Flu according to rapid tests. Managed to get a few things done through sheer force of will. I don't recommend it. Red Hat A lot of misdirection related to our international move towards France where my wife has family. The policies at Red Hat may mean that we need to come up with a strategy to keep a million-plus lines of code maintained if my only option to protect my family is to go the route of long-stay-visa (e.g. work not permitted). Some very lovely people have reached out and thank you for that! Touch base if you have options that would allow me to work from France if you like my approach to engineering. While I'd love to continue working on GNOME/GTK and related technologies, some things are more important and so I'm flexible. Lets all hope that the Corporate Leadership Team at Red Hat determines that supporting multi-ethnic families get safely out of the United States near their support network is of value to our mutual success. Ptyxis Triage a handful of issues. This project has a pretty high-velocity file-rate while having an pretty low "bug exists here" rate. This legitimately burns a lot of my time every week that could be spent creating things. So if you are going to file issues in projects like Ptyxis or Text Editor, please take the time to do basic binary-search on where the problem really exists. Points given just for trying. In fact, this is a rant about how you should do that in all aspects of your life. Cut the problems you have in half — and half again. Libdex Fumbled my way through adding a gi override to integrate DexFuture with asyncio. This allows for await some_future() but you need to make sure you have a GLib main context running as well and integrated with asyncio. Hopefully this means that you will soon be able to easily use all the nice Foundry APIs from Python with relative ease. Foundry Add support for communicating with SSH agent for simple signing requests. Was surprised to not find this in libssh2 but maybe I missed something. We already link against that for libgit2 support. Add support for signing commits in the FoundryGitCommitBuilder commit helper. This supports GPG and SSH commit signing (no X509) because I have no need for it. Though the SSH form is tested best. Iteration on tracking changes to untracked/unstaged/staged files while using the commit builder API. Abstract commit signing into generic buffer signing so that it can be used for more things in the future (like signing tags). Add support for staging/unstaging files, hunks, and lines. New test-git-commit-builder-gtk to be able to test out the infrastructure for commit creation. It's extremely scrappy but gets the job done enough to test the API out even if not very ergonomic. New foundry_git_vcs_stash() helper to be like git stash. Lots of new APIs around generating diffs, deltas, and patches. Lots of refactoring of Git subsystem to make the combination of threading and convenient APIs easier to maintain. Builder Some work on git change management panel, diff viewer, etc VCS history panel which is useful as auxiliary for files as well as for viewing diffs. Details auxiliary panel for forge issues/merge-requests Experiment with using AdwPreferencesGroup and friends for creating AdwBottomSheet style menus in narrow mode (e.g. mobile).
- Federico Mena-Quintero: Mutation testing for librsvg (2025/12/01 13:06)
  I was reading a blog post about the testing strategy for the Wild linker, when I came upon a link to cargo-mutants, a mutation testing tool for Rust. The tool promised to be easy to set up, so I gave it a try. I'm happy to find that it totally delivers! Briefly: mutation testing catches cases

where bugs are deliberately inserted in the source code, but the test suite fails to catch them: after making the incorrect changes, all the tests still pass. This indicates a gap in the test suite. Previously I had only seen mentions of "mutation testing" in passing, as something exotic to be done when testing compilers. I don't recall seeing it as a general tool; maybe I have not been looking closely enough. Setup and running Setting up cargo-mutants is easy enough: you can cargo install cargo-mutants and run it with cargo mutants. For librsvg this ran for a few hours, but I discovered a couple of things related to the way the librsvg repository is structured. The repo is a cargo workspace with multiple crates: the librsvg implementation and public Rust API, the rsvg-convert binary, and some utilities like rsvg-bench. By default cargo-mutants only seemed to pick up the tests for rsvg-convert. I think it may have done this because it is the only binary in the workspace that has a test suite (e.g. rsvg-bench does not have a test suite). I had to run cargo mutants --package librsvg to tell it to consider the test suite for the librsvg crate, which is the main library. I think I could have used cargo mutants --workspace to make it run all the things; maybe I'll try that next time. Initial results My initial run on rsvg-covert produced useful results; cargo-mutants found 32 mutations in the rsvg-convert source code that ought to have caused failures, but the test suite didn't catch them. The second run, on the librsvg crate, took about 10 hours. It is fascinating to watch it run. In the end it found 889 mutations with bugs that the test suite couldn't catch: 5243 mutants tested in 9h 53m 15s: 889 missed, 3663 caught, 674 unviable, 17 timeouts What does that mean? 5243 mutants tested: how many modifications were tried on the code. 889 missed: The important ones: after a modification was made, the test suite failed to catch this modification. 3663 caught: Good! The test suite caught these! 674 unviable: These modifications didn't compile. Nothing to do. 17 timeouts: Worth investigating; maybe a function can be marked to be skipped for mutation. Starting to analyze the results Due to the way cargo-mutants works, the "missed" results come in an arbitrary order, spread among all the source files: rsvg/src/path_parser.rs:857:9: replace <impl fmt::Display for ParseError>::fmt -> fmt::Result with Ok(Default::default()) rsvg/src/drawing_ctx.rs:732:33: replace > with == in DrawingCtx::check_layer_nesting_depth rsvg/src/filters/lighting.rs:931:16: replace / with * in Normal::bottom_left rsvg/src/test_utils/compare_surfaces.rs:24:9: replace <impl fmt::Display for BufferDiff>::fmt -> fmt::Result with Ok(Default::default()) rsvg/src/filters/turbulence.rs:133:22: replace - with / in setup_seed rsvg/src/document.rs:627:24: replace match guard is_mime_type(x, "image", "svg+xml") with false in ResourceType::from rsvg/src/length.rs:472:57: replace * with + in CssLength<N, V>::to_points So, I started by sorting the missed.txt file from the results. This is much better: rsvg/src/accept_language.rs:136:9: replace AcceptLanguage::any_matches -> bool with false rsvg/src/accept_language.rs:136:9: replace AcceptLanguage::any_matches -> bool with true rsvg/src/accept_language.rs:78:9: replace <impl fmt::Display for AcceptLanguageError>::fmt -> fmt::Result with Ok(Default::default()) rsvg/src/angle.rs:40:22: replace < with <= in Angle::bisect rsvg/src/angle.rs:41:56: replace - with + in Angle::bisect rsvg/src/angle.rs:49:35: replace + with - in Angle::flip rsvg/src/angle.rs:57:23: replace < with <= in Angle::normalize With the sorted results, I can clearly see how cargo-mutants gradually does its modifications on (say) all the arithmetic and logic operators to try to find changes that would not be caught by the test suite. Look at the first two lines from above, the ones that refer to AcceptLanguage::any_matches: rsvg/src/accept_language.rs:136:9: replace AcceptLanguage::any_matches -> bool with false rsvg/src/accept_language.rs:136:9: replace AcceptLanguage::any_matches -> bool with true Now look at the corresponding lines in the source: ... impl AcceptLanguage { 135 fn any_matches(&self, tag: &LanguageTag) -> bool { 136 self.iter().any(|(self_tag, _weight)| tag.matches(self_tag)) 137 } ... } } The two lines from missed.txt mean that if the body of this any_matches() function were replaced with just true or false, instead of its actual work, there would be no failed tests: 135 fn any_matches(&self, tag: &LanguageTag) -> bool { 136 false // or true, either version wouldn't affect the tests 137 } } This is bad! It indicates that the test suite does not check that this function, or the surrounding code, is working correctly. And yet, the test coverage report for those lines shows that they are

indeed getting executed by the test suite. What is going on? I think this is what is happening: The librsvg crate's tests do not have tests for AcceptLanguage::any_matches. The rsvg_convert crate's integration tests do have a test for its --accept-language option, and that is what causes this code to get executed and shown as covered in the coverage report. This run of cargo-mutants was just for the librsvg crate, not for the integrated librsvg plus rsvg_convert. Getting a bit pedantic with the purpose of tests, rsvg-convert assumes that the underlying librsvg library works correctly. The library advertises support in its API for matching based on AcceptLanguage, even though it doesn't test it internally. On the other hand, rsvg-convert has a test for its own --accept-language option, in the sense of "did we implement this command-line option correctly", not in the sense of "does librsvg implement the AcceptLanguage functionality correctly". After adding a little unit test for AcceptLanguage::any_matches in the librsvg crate, we can run cargo-mutants just for that the accept_language.rs file again: # cargo mutants --package librsvg --file accept_language.rs Found 37 mutants to test ok Unmutated baseline in 24.9s build + 6.1s test INFO Auto-set test timeout to 31s MISSED rsvg/src/accept_language.rs:78:9: replace <impl fmt::Display for AcceptLanguageError>::fmt -> fmt::Result with Ok(Default::default()) in 4.8s build + 6.5s test 37 mutants tested in 2m 59s: 1 missed, 26 caught, 10 unviable Great! As expected, we just have 1 missed mutant on that file now. Let's look into it. The function in question is now <impl fmt::Display for AcceptLanguageError>::fmt, an error formatter for the AcceptLanguageError type: impl fmt::Display for AcceptLanguageError { fn fmt(&self, f: &mut fmt::Formatter<'_>) -> fmt::Result { match self { Self::NoElements => write!(f, "no language tags in list"), Self::InvalidCharacters => write!(f, "invalid characters in language list"), Self::InvalidLanguageTag(e) => write!(f, "invalid language tag: {e}"), Self::InvalidWeight => write!(f, "invalid q= weight"), } } } What cargo-mutants means by "replace ... -> fmt::Result with Ok(Default::default()) is that if this function were modified to just be like this: impl fmt::Display for AcceptLanguageError { fn fmt(&self, f: &mut fmt::Formatter<'_>) -> fmt::Result { Ok(Default::default()) } } then no tests would catch that. Now, this is just a formatter function; the fmt::Result it returns is just whether the underlying call to write!() succeeded. When cargo-mutants discovers that it can change this function to return Ok(Default::default()) it is because fmt::Result is defined as Result<(), fmt::Error>, which implements Default because the unit type () implements Default. In librsvg, those AcceptLanguageError errors are just surfaced as strings for rsvg-convert, so that if you give it a command-line argument with an invalid value like --accept-language=foo, it will print the appropriate error. However, rsvg-convert does not make any promises as to the content of error messages, so I think it is acceptable to not test this error formatter — just to make sure it handles all the cases, which is already guaranteed by its match statement. Rationale: There already are tests to ensure that the error codes are computed correctly in the parser for AcceptLanguage; those are the AcceptLanguageError's enumeration variants. There is a test in rsvg-convert's test suite to ensure that it detects invalid language tags and reports them. For cases like this, cargo-mutants allows marking code to be skipped. After marking this fmt implementation with #[mutants::skip], there are no more missed mutants in accept_language.rs. Yay! Understanding the tool You should absolutely read "using results" in the cargo-mutants documentation, which is very well-written. It gives excellent suggestions for how to deal with missed mutants. Again, these indicate potential gaps in your test suite. The documentation discusses how to think about what to do, and I found it very helpful. Then you should read about genres of mutants. It tells you the kind of modifications that cargo-mutants does to your code. Apart from changing individual operators to try to compute incorrect results, it also does things like replacing whole function bodies to return a different value instead. What if a function returns Default::default() instead of your carefully computed value? What if a boolean function always returns true? What if a function that returns a HashMap always returns an empty hash table, or one full with the product of all keys and values? That is, do your tests actually check your invariants, or your assumptions about the shape of the results of computations? It is really interesting stuff! Future work for librsvg The documentation for cargo-mutants

suggests how to use it in CI, to ensure that no uncaught mutants are merged into the code. I will probably investigate this once I have fixed all the missed mutants; this will take me a few weeks at least. Librsvg already has the gitlab incantation to show test coverage for patches in merge requests, so it would be nice to know if the existing tests, or any new added tests, are missing any conditions in the MR. That can be caught with cargo-mutants. Hackery relevant to my tests, but not to this article If you are just reading about mutation testing, you can ignore this section. If you are interested in the practicalities of compilation, read on! The source code for the librsvg crate uses a bit of conditional compilation to select whether to export functions that are used by the integration tests as well as the crate's internal tests. For example, there is some code for diffing two images, and this is used when comparing the pixel output of rendering an SVG to a reference image. For historical reasons, this code ended up in the main library, so that it can run its own internal tests, but then the rest of the integration tests also use this code to diff images. The librsvg crate exports the "diff two images" functions only if it is being compiled for the integration tests, and it doesn't export them for a normal build of the public API. Somehow, cargo-mutants didn't understand this, and the integration tests did not build since the cargo feature to select that conditionally-compiled code... wasn't active, or something. I tried enabling it by hand with something like cargo mutants --package librsvg ---features test-utils but that still didn't work. So, I hacked up a temporary version of the source tree just for mutation testing, which always exports the functions for diffing images, without conditional compilation. In the future it might be possible to split out that code to a separate crate that is only used where needed and never exported. I am not sure how it would be structured, since that code also depends on librsvg's internal representation of pixel images. Maybe we can move the whole thing out to a separate crate? Stop using Cairo image surfaces as the way to represent pixel images? Who knows!

- Sophie Herold: Weekly report #75 (2025/11/30 21:03)
  Hello world! Last week, I asked the people that financially support me, if I should post my updates publicly. A majority voted to release my future weekly reports to the public, some voted to make them public every other week. So I will try to post some of them publicly in the future. These updates are made possible by the amazing people that support me on Ko-fi, Patreon, Open Collective, and GitHub! Thank you so much, folks! Since this is my first public weekly report, let’s maybe start with a short introduction: I started my GNOME related work in 2018 by working a bit on Gajim UI and starting the Pika Backup project. Since March 2024 I have slowly started to ask for donations for my work on GNOME. I am disabled due to ME/CFS and being autistic. Working within the GNOME project allows me to earn a bit of extra money on top of my social assistance while also doing something that I love, and I can do at my own pace. I am working on too many things within GNOME: Pika Backup, Loupe, glycin, websites, Release Team, Circle Committee, and a bunch of other things, like trying to advocate for queer and disabled people within the GNOME community. You will notice that my weekly reports will usually not contain giant achievements or huge promises. Maintenance work can be tedious, and generally, fundraisers have developed a frustrating habit of frequently over-promising and under-delivering. I don’t want to be part of that. So, let’s finally get to the actual updates for last week. We landed the translation support within www.gnome.org. At the moment, this is still practically invisible. We are waiting for the Translation Team to enable translators to do the work. Once we got some translations, we will also enable the language selection dialog. I also asked if we want translations for donate.gnome.org, but got no feedback so far. A release for gst-thumbnailers is now out, such that distributions can package it. There is more about the thumbnailers in the Release Team issue. I updated the Circle benefits list, and updated circle.gnome.org to the version that does not list apps and components on its own. That’s something design people wanted to see for a while. Since the old Circle page stopped building, that was a good moment to finally do it :) I spend some time on Pika Backup hoping that we are very close to a 0.8 beta release. However, I noticed that the current state of the setup dialog isn’t

what we want. After discussing the options with Fina, we are now sure that we have to rework what we have in some way. Not shying away from throwing code away, and reworking something again, is often very important for approaching at a good result. Maybe I will summarize this example, once we have arrived at a solution. Some of the less tangible work this week: Shortly discussed Emmanuele's GNOME Governance proposal in Matrix. Something that might look like making changes within GNOME more complicated from the outside. But the actual goal is the opposite: Currently, it can be very hard to make changes within GNOME since there is no clear way how to go about it. This not only slows people down but, at least for me, can also me quite emotionally draining. So, a very important proposal. Maybe we got a tiny step closer to making it reality. Also contributed to an internal Release Team discussion and contacted an involved party. That's all for this week! If you want to support my work financially, you can check my GitLab profile. Hope you all have a great week!

- This Week in GNOME: #227 Circle Benefits (2025/11/29 00:00)
  Update on what happened across the GNOME project in the week from November 22 to November 29. GNOME Circle Apps and Libraries Sophie (she/her) says The benefits for GNOME Circle projects now explicitly include the participation in internship programs as well as the inclusion on the help.gnome.org, welcome.gnome.org, apps.gnome.org or developer.gnome.org pages. The circle.gnome.org page as been redesigned to link to the respective pages instead of having its own app and component list. NewsFlash feed reader ↗ Follow your favorite blogs & news sites. Jan Lukas reports Last week Newsflash 4.2 was released with the usual amount of small improvements and fixes. What makes it worthy of the minor version bump in my opinion is the new popover right above the article containing all relevant font and spacing options for a more direct interaction, a new slightly different layout for tablets (specifically the PineNote) and combining multiple image enclosures into a carousel. Third Party Projects Alain announces Planify 4.16.1 is now available! A small but polished update focused on improving the overall experience: • UX improvements and minor bug fixes • Shift+Enter is back for quick "keep adding" • Zoom links now supported in calendar events • Updated Donate page • The Planify website has been refreshed with a cleaner design and several improvements → https://www.useplanify.com Thanks for following the journey of Planify Dzheremi announces Mass Lyrics Downloading with Chronograph 5.3 Chronograph got an update with, as promised before, mass lyrics downloading support. Now users allowed to query LRClib to give them lyrics for all tracks in their current library. Chronograph will try to find the closest lyrics among results for your tracks. This feature expands coverage of the app usefulness for users, who doesn't want to sync lyrics themselves, but want to just download them. Sync lyrics of your loved songs Parabolic ↗ Download web video and audio. Nick announces Parabolic V2025.11.1 is here! This release contains many bug fixes for issues users were experiencing. A larger new feature update is in the works to address many long standing feature requests. Here's the full changelog: Fixed the sleep interval for multiple subtitle downloads Fixed an issue where low-resolution media was being downloaded on Windows Fixed an issue where aria2c couldn't download media from certain sites Fixed an issue where Remove Source Data was not clearing all identifiable metadata fields Shell Extensions boerdereinar announces Hey everyone, this week I've released my clipboard manager extension Copyous with the following features: Supports text, code, images, files, links, characters and colors. Can be opened at mouse pointer or text cursor Pin favorite items Group items with 9 colored tags Customizable clipboard actions Highly customizable PakoVM announces This week I publish Tinted Shell, a extension that simply adds a spalsh of color to the Gnome Shell theme based on the user's current accent color while respecting the original look. You can get it from Gnome Extensions and contribute to it on Github. Miscellaneous Krafting - Vincent says Hey everyone, this week I pushed updates to use the GNOME 49 runtime, and revamping the keyboard shortcuts pages on all my apps available on Flathub : PedantiK SemantiK Reddy Playlifin Playlifin Voyager Hex Colordle In addition, SemantiK got some versions bumps and PedantiK got a lot of bug fixes (including fixing the broken

Wikipedia API) Also, work started on a PedantiK English pack, to allow languages other than french. GNOME Foundation Allan Day announces A new GNOME Foundation update is available, covering what has happened at the GNOME Foundation over the past two weeks. It covers a fairly long list of topics, including the recent budget report, funding for Outreachy, banking and finance changes, Flathub progress, and more. Digital Wellbeing Project ↗ Philip Withnall reports This week in parental controls, Ignacy is working on changing the Shell lock screen to show when a child's screen time limit has been reached; and I've spent a bit of time writing up the technical details of how web filtering will work at https://tecnocode.co.uk/2025/11/27/parental-controls-web-filtering-backend/ (the backend is written, the UI integration is future work) 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!

- Christian Hergert: Libdex Futures from asyncio (2025/11/28 21:18)
  One of my original hopes for Libdex was to help us structure complex asynchronous code in the GNOME platform. If my work creating Foundry and Sysprof are any indicator, it has done leaps and bounds for my productivity and quality in that regard. Always in the back of my mind I hoped we could make those futures integrate with language runtimes. This morning I finally got around to learning enough of Python's asyncio module to write the extremely minimal glue code. Here is a commit that implements integration as a PyGObject introspection override. It will automatically load when you from gi.repository import Dex. This only integrates with the asyncio side of things so your application is still responsible for integrating asyncio and GMainContext or else nothing will be pumping the DexScheduler. But that is application toolkit specific so I must leave that to you. I would absolutely love it if someone could work on the same level of integration for GJS as I have even less experience with that platform.
- Allan Day: GNOME Foundation Update, 2025-11-28 (2025/11/28 19:26)
  Welcome to another GNOME Foundation update; an overview of everything that's been happening at the Foundation. There was no update last week, due to me being away from my computer last Friday, so this post covers a two week period rather than the usual single week. Many thanks to everyone who responded to my request for feedback in my previous post! It was great to hear your views on these posts, and it was extremely motivating to get positive feedback on the blog series. Budget report In case you didn't see it, last week Rob posted a detailed breakdown of the Foundation's current operating budget. This is the second year in a row that we have provided a budget report for the community, and I'm thrilled that we've been able to keep up the momentum around financial transparency. I'd encourage you to give it a read if you haven't already. Community travel One positive aspect of the current budget is that we have a healthy community travel budget, and I really want to encourage members to make use of the fund. The travel budget is there to be spent, and we absolutely want to see community members applying for travel. If you have been interested in organising a hackfest, or attending a GNOME conference, and finances have been a barrier, please do make use of the funding that is available. Information about how to apply can be found in the handbook. Also on travel: we are currently looking to recruit additional volunteers to help administer the travel budget, as part of the Travel Committee. So, if you are interested in helping with GNOME and would like to get involved, please do get in touch using the comments below, or by messaging the Travel Committee. Outreachy The Foundation has a proud history of funding outreach efforts, and has regularly supported interns through Outreachy. The December to March round is almost upon us, and the Internship Committee has coordinated the selection of an intern who we will be sponsoring. We were pleased to release the funding for this internship this week. More details about the internship itself will follow. Banking and finance systems As mentioned in recent updates, we have been working through a round of improvements to our banking setup, which will give us enhanced fraud protection, as well as automatic finance management features. This week we had a training session with our bank, the fraud protection features were turned

on, and I signed the last of the paperwork. As a result, this round of work is now complete. I have also been going through the process of signing up for the new financial system that Dawn our new finance advisor will be setting up for us. Bookkeeping meetings Our regular monthly bookkeeping meeting happened last week, and we had another follow-up call more recently. We are still working through the 2024-25 financial year end accounts, which primarily involves resolving a list of small questions, to make sure that the accounts are 100% complete and accurate. Our bookkeeper has also been very patiently answering questions from Deepa, our treasurer, and myself as we continue to familiarise ourselves with the finance and accounting setup (thank you!) Board meeting The Board had a regular meeting this week. The topics under discussion included: Setting goals for the upcoming fundraising campaign, in particular what the fundraising target should be, and what programs we want to fund with the proceeds. Improving our minutes to meet the needs of different audiences (directors, auditors, the IRS, members, and so on). We also worked on a plan to clear the backlog of unapproved minutes. Planning for a Board hackfest prior to next FOSDEM. We came away with a healthy list of action items, and I'm looking forward to making progress in each of these areas. GNOME.Asia Our upcoming conference is Tokyo continues to be a focus, and Kristi is busy putting the final arrangements together. The event is just 15 days away! A reminder: if you want to participate, please do head over to the site and register. Flathub There has been some good progress around Flathub over the past two weeks. Bart has done quite a bit of work to improve the performance of the Flathub website, which I'm sure users will appreciate. We also received some key pieces of legal work, which are required as part of the roadmap to establish Flathub as its own financial/legal entity. With those legal documents in place we have turned our attention to planning Flathub's financial systems; discussions about this are ongoing. Digital Wellbeing There was another review call this week to check on progress as the current phase of the program reaches its final stages. The main focus right now is making sure that the new screen time limits feature is in good shape before we use up the remaining funding. Progress is looking good in general: the main changes for GNOME Shell and Settings have all been merged. There are two more pieces of work to land before we can say that we are in a feature complete state. After that we will circle back to UX review and papercut fixing. If you want more information about these features, I would recommend Ignacy's recent post on the topic. Philip has also published a fantastic post on the web filtering functionality that has been implemented as part of this program. That's it for this week! Thanks for reading, and see you next week.

- Philip Withnall: Parental controls web filtering backend (2025/11/27 13:25)
  In my previous post I gave an overview of the backend for the screen time limits feature of parental controls in GNOME. In this post, I'll try and do the same for the web filtering feature. We haven't said much about web filtering so far, because the user interface for it isn't finished yet. The backend is, though, and it will get plumbed up eventually. Currently we don't have a GNOME release targeted for it yet. When is web filterings? What is web filtering? (Apologies to Radio 4 Friday Night Comedy.) Firstly, what is the aim of web filtering? As with screen time limits, we've written a design document which (hopefully) covers everything. But the summary is that it should allow parents to filter out age-inappropriate content on the web when it's accessed by child accounts, while not breaking the web (for example, by breaking TLS for websites) and not requiring us (as a project) to become curators of filter lists. It needs to work for all apps on the system (lots of apps other than web browsers can show web content), and needs to be able to filter things differently for different users (two different children of different ages might use the same computer, as well as the parents themselves). After looking at various different possible ways of implementing it, the best solution seemed to be to write an NSS module to respond to name resolution (i.e. DNS) requests and potentially block them according to a per-user filter list. A brief introduction to NSS NSS (Name Service Switch) is a standardised name lookup API in libc. It's used for hostname resolution, but also for user accounts and various other things. Names are resolved by various modules which are dlopen()ed into your process by libc and queried in the

order given in /etc/nsswitch.conf. So for hostname resolution, a typical configuration in nsswitch.conf would cause libc to query the module which looks at /etc/hosts first, then the module which checks your machine's hostname, then the mDNS module, then systemd-resolved. So, we can insert our NSS module into /etc/nsswitch.conf, have it run somewhere before systemd-resolved (which in this example does the actual DNS resolution), and have it return a sinkhole address for blocked domains. Because /etc/nsswitch.conf is read by libc within your process, this means that the configuration needs to be modified for containers (flatpak) as well as on the host system. Because the filter module is loaded into the name lookup layer, this means that content filtering (as opposed to domain name filtering) is not possible with this approach. That's fine — content filtering is hard, I'm not sure it gives better results overall than domain name filtering, and means we can't rely on existing domain name filter lists which are well maintained and regularly updated. We're not planning on adding content filtering. It also means that DNS-over-HTTPS/-TLS can be supported, as long as the app doesn't implement it natively (i.e. by talking HTTPS over a socket itself). Some browsers do that, so the module needs to set a canary to tell them to disable it. DNS-over-HTTPS/-TLS can still be used if it's implemented by one of the NSS modules, like systemd-resolved. Nothing here stops apps from deliberately bypassing the filtering if they want, perhaps by talking DNS over UDP directly, or by calling secret internal glibc functions to override nsswitch.conf. In the future, we'd have to implement per-app network sandboxing to prevent bypasses. But for the moment, trusting the apps to cooperate with parental controls is fine. Filter update daemon So we have a way of blocking things; but how does it know what to block? There are a lot of filter lists out there on the internet, targeted at existing web filtering software. Basically, a filter list is a list of domain names to block. Some filter lists allow wildcards and regexps, others just allow plain strings. For simplicity, we've gone with plain strings. We allow the parent to choose zero or more filter lists to build a web filtering policy for a child. Typically, these filter lists will correspond to categories of content, so the parent could choose a filter list for advertising, and another for violent content, for example. The web filtering policy is basically the set of these filter lists, plus some options like "do you want to enforce safe search". This policy is, like all other parental controls policies, stored against the child user in accounts-service. Combine these filter lists, and you have the filter list to give to NSS in the child's session, right? Not quite — because the internet unfortunately keeps changing, filter lists need to be updated regularly. So actually what we need is a system daemon which can regularly check the filter lists for updates, combine them, and make them available as a compiled file to the child's NSS module — for each user on the system. This daemon is malcontent-webd. It has a D-Bus interface to allow the parent to trigger compiling the filter for a child when changing the parental controls policy for that child in the UI, and to get detailed feedback on any errors. Since the filter lists come from third parties on the internet, there are various ways they could have an error. It also has a timer unit trigger, malcontent-webd-update, which is what triggers it to periodically check the filter lists for all users for updates. High-level diagram of the web filtering system, showing the major daemons and processes, files, and IPC calls. If it's not clear, the awful squiggled line in the bottom left is meant to be a cloud. Maybe this representation is apt. And that's it! Hopefully it'll be available in a GNOME release once we've implemented the user interface for it and done some more end-to-end testing, but the screen time limits work is taking priority over it.

- Sam Thursfield: Bollocks to Github (2025/11/27 00:27)
  I am spending the evening deleting my Github.com account. There are plenty of reasons you might want to delete your Github account. I'd love to say that this is a coherently orchestrated boycott on my part, in sympathy with the No Azure for Apartheid movement. Microsoft, owner of Github, is a big pile of cash happy to do business with an apartheid state. That's a great reason to delete your Github.com account. I will be honest with you though, the thing that pushed me over the edge was a spam email they sent entitled "GitHub Copilot: What's in your free plan ". I was in a petty mood this morning. Offering free LLM access is a money loser. The long play is this: Microsoft would like to create a generation of

computer users hooked on GitHub Copilot. And, I have to hand it to them, they have an excellent track record in monopolising how we interact with our PCs. Deleting my Github.com account isn't going to solve any of that. But it feels good to be leaving, anyway. The one billionth Github repository was created recently and it has a single line README containing the word "shit". I think that summarizes the situation more poetically than I could. I had 145 repositories in the ssssam/ namespace to delete. The oldest was Iris; forked in 2011. Quite a story to that one. A fork of a project by legendary GNOME hacker Christian Hergert. In early 2011, I'd finished university, had no desire to work in the software industry, and was hacking on a GTK based music player app from time to time. A rite of passage that every developer has to go through, I suppose. At some point I decided to overcomplicate some aspect of the app and ended up integrating libiris, a library to manage concurrent tasks. Then I started working professionally and abandoned the thing. Its fun to look at that with 13 years perspective. I since learned, largely thanks to Rust, that I cannot possibly ever write correct concurrent thread-based C code. (All my libiris changes had weird race conditions). I met Christian various times. Christian created libdex which does the same thing, but much better. I revived the music player app as a playlist generation toolkit. We all lived happily ever after. Except for the Github fork, which is gone. What else? This guy was an early attempt at creating a sort of GObject mapping for SPARQL data as exposed by Localsearch (then Tracker). Also created around 2011. Years later, I did a much better implementation in TrackerResource which we still use today. The Sam of 2011 would be surprised to hear that we organized GUADEC in Manchester only 6 years later. Back in those days we for some reason maintained our own registration system written in Node.js. I spent the first few weeks of 2017 hacking in support for accommodation bookings. I discovered another 10 year old gem called "Software Integration Ontology." Nowadays we'd call that an SBOM. Did that term exist 10 years ago? I have spent too much time working on software integration. Various other artifacts research into software integration and complexity. A vestigal "Software Dependency Visualizer" project. (Which took on a life of its own, and many years later the idea is alive in KDE Codevis). A fork of Aboriginal Linux, which we unwittingly brought to an end back in 2016. Bits of Baserock, which never went very far but also led to the beginning of BuildStream. A fork of xdg-app, which is the original name of Flatpak. A library binding GLib to the MS RPC API on Windows, from the 2nd professional project I ever did. These things are now dust. I had over 100 repositories on github.com. I sponsored one person, who I can't sponsor any more as they only accept Github money. (I sponsor plenty of people in other platforms) Anyway, lots of nice people use Github, you can keep using Github. I will probably have to create the occasional burner account to push PRs where projects haven't migrated away. Some of my projects are now in Gitlab.com and in GNOME's Gitlab. Others are gone! It's a weird thing but in 2025 I'm actually happy knowing that there's a little bit less code in the world.

- Michael Meeks: <h1>a new & beautiful Collabora Office</h1> (2025/11/26 21:00)
  Just a short personal note to say how super excited I am to get our very first release of a new Collabora Office out that brings Collabora Online's lovely UX - created by the whole team to the desktop. You can read all about it in the press release. Please note - this is a first release - we expect all manner of unforseen problems, but still - it edits documents nicely. The heros behind the scenes There has been a huge amount of work behind the scenes, and people to say thank-you to. Let me try to get some of them: First off - thanks to Jan 'Kendy' Holesovsky and Tor Lillqvist (who came out of retirement to create yet another foundational heavy-lift for FLOSS Office. I can't say how grateful we are for your hard work here on Mac and Windows respectively. Then after the allotropia merger we had Thorsten Behrens to lead the project, and Sarper Akdemir to drive the Linux front-end. Towards the end of the project we were thrilled to expand things to include a dream-team of FLOSS engineers to fix bugs and add features ... Thanks to Rashesh Padia for the lovely first-start WebGL slideshow presentation added to Richard Brock's content skills. Thanks to Vivek Javiya for building a new file creation UI with Pedro Silva's design skills here and elsewhere. Lots of bug fix and polishing work

from Parth Raiyani, and Jeremy Whiting (who also did multi-tabbed interface on Mac), and to Stephan Bergmann for digging out and clobbering the most hard-core races and horror bugs that we had hidden, Caolán McNamara too who made multiple documents work, and fixed crashes and multi-screen bits. With Hubert Figuière making the flatpak beautiful, and of course the indomitable Andras Timar doing so much amazing work getting all of the CI, release-engineering, app-store, translation pieces and also bug-fixing done and completed in time. Thanks too to our marketing team: from Chris Thornett getting the press briefing into a good state and multiplexing quotes left and right, to Richard Brock creating beautiful blog output, to Asja Čandić socializing it all, with Naomi Obbard leading the charge. Thanks also to Darshan Upadhyay for packing the community website with hard-working-beaver goodness, updated community page, as well as new build instructions, FAQ, supported pages, and much more. Thanks to all of our supporters who say nice things about us, and of course to so many translators who contribute to making Collabora Online great - hopefully now the strings are all public it should be easy to expand coverage. This is an outstanding result from so many - thank you! What is next technically ? There are lots of things we plan to do next, but there is so much that can be done. First - merging the work into our main product branches - and at the same time sharing much more of the code across platforms. We have some features in the pipeline already - starting to take more advantage of platform APIs for much improved slideshow / multi-screen presentation pieces that need merging and releasing, and ultimately better printing APIs, and better copy/paste. Then we need to make sure that all of the new features are present on all platforms - multi-tabbed UI, the new file creation UI, and of course much more polish and bug fixing - as well as better automated testing. Its exciting to have a big new release - but in general - we work really quite hard to avoid having big-bang deadlines, and a more steady development cadence. We will be trying to get the feature conveyer belt working alongside the Collabora Online development process - reasonably quickly - but now with another three platforms. Then over the next months - there are various fairly obvious directions to take the code in - one amusing feature was the ability to apparently collaborate with yourself when loading the same document on the same machine, that can be extended. Conclusion It has been really exciting to get feedback from many partners, customers and community members about the need for this. Again - this is a very first release - we plan to do lots of iteration and improvement around it. But, thank you again to the whole team and community for making Collabora Online something that people really want us to bring to the desktop. If you'd like to get involved (or just see pretty artwork of hard working animals) - why not head to community website, or our forum or code. Rock on =)

- Sam Thursfield: Status update — 23/11/2025 (2025/11/25 00:03)
  Bo día. I am writing this from a high speed train heading towards Madrid, en route to Manchester. I have a mild hangover, and a two hundred page printout of “The STPA Handbook”… so I will have no problem sleeping through the journey. I think the only thing keeping me awake is the stunning view. Sadly I havent got time to go all the way by train, in Madrid i will transfer to Easy Jet. It is indeed easy compared to trying to get from Madrid into France by train. Apparently this is mainly the fault of France’s SNCF. On the Spain side, fair play. The ministro de fomento (I think this translates as “Guy in charge of trains?”) just announced major works in Barcelona, including a new station in La Sagrera with space for more trains than they have now, and a more direct access from Madrid, and a speed boost via some new type of railway sleeper, which would make the top speed 350km/h instead of 300km/h as it is now. And some changes in Madrid, which would reduce the transfer time when arriving from the west and heading out further east. You can argue with many things about the trains in Spain… perhaps it would be useful if the regional trains here ran more than once per day… but you cant argue with the commitment to fast inter-city travel. If only we had similar investment to fix the cross border links between Spain and France, which are something of a joke. Engineers around the world will know this story. The problem is social: two different organizations, who speak different languages, have to agree on something. There is already a perfectly usable modern

train line across the border. How many trains per day? Uh… two. Hope you planned your trip in advance because they're fully booked next week. Anyway, this isn't meant to be a post on the status of the railways of western Europe. Digital Resilience Forum Last month I hopped on another Madrid-bound train to attend the Digital Resilience Forum. It's a one day conference organized by Bitergia who you might know as world leaders in open source community analysis. I have mixed feelings about "community metrics" projects. As Nick Wellnhofer said regarding libxml, when you participate as a volunteer in a project that is being monitored, its easy to feel like you're being somehow manipulated by the corporations who sponsor these things. How come you guys will spend time and money analyzing my project's development processes and Git history, but you won't spend time actually fixing bugs and improvements upstream? As the ffmpeg developers said: How come you will pay top calibre security researchers to read our code and find very specific exploits, but then wait for volunteers to fix them? The Bitergia team are great people who genuinely care about open source, and I really enjoyed the conference. The main themes were: digital sovereignty, geopolitics, the rise of open source, and that XKCD where all our digitalinfrastructure depends on a single unpaid volunteer in Nebraska. (https://xkcd.com/2347/). (Coincidentally, one of the Bitergia guys actually does live in Nebraska). It was a day in a world where I am not used to participating: less engineering, more politics and campaigning. Yes, the Sovereign Tech Agency were around. We played a cool role play game simulating various hypothetical software crisis that might happen in the year 2027 (spoiler: in most cases a vendor-neutral, state-funded organization focused on open source was able to save the day : -). It is amazing what they've done so far with a relatively small investment, but it is a small organization and they maintain that citizens of every country should be campaigning and organizing to setting up an equivalent. Let's not tie the health of open source infrastructure too closely to German politics. Also present, various campaign groups with "Open" at the start of their name: OpenForum Europe, OpenUK, OpenIreland, OpenRail. When I think about the future of Free Software platforms, such as our beloved GNOME, my mind always goes to funding contributors. There's very little money here and meanwhile Apple and Microsoft have nearly all of the money and I feel like still GNOME succeeds largely thanks to the evenings and weekends of a small core of dedicated hackers; including some whose day job involves working on some other part of GNOME. It's a bit depressing sometimes to see things this way, because the global economy gets more unequal every day, and how do you convince people who are already squeezed for cash to pay for something that's freely available online? How do you get students facing a super competitive job market to hack on GTK instead of studying for university exams? There's another side which I talk about less, and that's education. There are more desktop Linux users than ever — apparently 5% of all desktop users or something — but there's still very little agreement or understanding what "open source" is. Most computer users couldn't tell you what an "operating system" does, and don't know why "source code" can be an interesting thing to share and modify. I don't like to espouse any dogmatic rule that the right way to solve any problem is to release software under the GPLv3. I think the problems society has today with technology come from over-complexity and under-study. (See also, my rant from last month.). To tackle that, it is important to have software infrastructure like drivers and compilers available under free software licenses. The Free Software movement has spent the last 40 years doing a pretty amazing job of that, and I think its surprising how widely software engineers accept that as normal and fight to maintain it. Things could easily be worse. But this effort is one part of a larger problem, of helping those people who think of themselves as "non-technical" to understand the fundamentals of computing and not see it as a magic box. Most people alive today have learned to read and write one or more languages, to do mathematics, to operate a car, to build spreadsheets, and operate a smartphone. Most people I know under 45 have learned to prompt a large language model in the last few years. With a basic grounding in how a computer operates, you can understand what an operating system does. And then you can see that whoever controls your OS has complete control over your digital life. And you will start to think twice about leaving that control to Apple,

Google and Microsoft — big piles of cash where the concept of “ethics” barely has a name. Reading was once a special skill reserved largely for monks. And it was difficult: we only started spaces between the words later on. Now everyone knows what a capital letter is. We need to teach how computers work, we need to stop making them so complicated, and the idea of open development will come into focus for everyone. (and yes i realize this sounds a bit like the permacomputing manifesto). Codethink work This is a long rant, isn’t it? My train only just left Zamora and I didnt fall asleep yet, so there’s more to come. I had a nice few months hacking on Endless OS 7, which has progressed from an experiment to a working system, bootable on bare metal, albeit with a various open issues that would block a stable release as yet. The overview docs in the repo tell you how to play with it. This is now fully in the hands of the team at Endless, and my next move is going to be in some internal research that has been ongoing for a number of years. Not much of it is secret, in fact quite a lot is being developed in the open, and it relates in part to regulatory compliance and safety-critical softare systems. Codethink dedicates more to open source than most companies its size. We never have trouble getting sponsorship for events like GUADEC. But I do wish I could spend more time maintaining open infrastructure that I use every day, like, you know, GNOME. This project isn’t going to solve that tomorrow, but it does occupy an interesting space in the intersection between industry and open source. The education gap I talked to you above is very much present in some industries where we work. Back in February a guy in a German car firm told me, “Nobody here wants open source. What they want is somebody to blame when the thing goes wrong.” Open source software comes with a big disclaimer that says, roughly, that if it breaks you get to keep both pieces. You get to blame yourself. And that’s a good thing! The people who understand a final, integrated system are the only people who can really define “correct behaviour”. If you’ve worked in the same industries I have you might recognise a common anti-pattern: teams who spend all their time arguing about ownership of a particular bug, and team A are convinced it’s a misbehaviour of component B and team B will try to prove the exact opposite. Meanwhile nobody actually spends the 15 minutes it would take to actually fix the bug. Another anti-pattern: team A would love to fix the bug in component B, but team B won’t let them even look at the source code. This happens muuuuuuuch more than you might think. So we’re not trying to teach the world how computers work, on this project, but we are trying to increase adoption and understanding at least in the software industry. There are some interesting ideas. Looking at software systems from new angles. This is where STPA comes in, by the way — it’s a way of breaking a system down not into components but rather into one or more control loops. Its going to take a while to make sense of everything in this new space… but you can expect some more 1500 word blog posts on the topic.

- Jussi Pakkanen: 3D models in PDF documents (2025/11/24 18:13)
  PDF can do a lot of things. One them is embedding 3D models in the file and displaying them. The user can orient them freely in 3D space and even choose how they should be rendered (wireframe, solid, etc). The main use case for this is engineering applications.Supporting 3D annotations is, as expected, unexpectedly difficult because:No open source PDF viewer seems to support 3D models.Even though the format specification is available, no open source software seems to support generating files in this format (by which I mean Blender does not do it by default). [1]But, again, given sufficient effort and submitting data to not-at-all-sketchy-looking 3D model conversion web sites, you can get 3D annotations to work. Almost.As you can probably tell, the picture above is not a screenshot. I had to take it with a cell phone camera, because while Acrobat Reader can open the file and display the result, it hard crashes before you can open the Windows screenshot tool. [1] Update: apparently KiCad nightly can export U3D files that can be used in PDFs.
- Jakub Steiner: 12 months instead of 12 minutes (2025/11/21 07:44)
  Hey Kids! Other than raving about GNOME.org being a static HTML, there’s one more aspect I’d like to get back to in this writing exercise called a

blog post. I've recently come across an apalling genAI website for a project I hold deerly so I thought I'd give a glimpse on how we used to do things in the olden days. It is probably not going to be done this way anymore in the enshittified timeline we ended up in. The two options available these days are — a quickly generated slop website or no website at all, because privately owned social media is where it's at. The wanna-be-catchy title of this post comes from the fact the website underwent numerous iterations (iterations is the core principle of good design) spanning over a year before we introduced the redesign. So how did we end up with a 3D model of a laptop for the hero image on the GNOME website, rather than something generated in a couple of seconds and a small town worth of drinking water or a simple SVG illustration? The hero image is static now, but used to be a scroll based animation at the early days. It could have become a simple vector style illustration, but I really enjoy the light interaction of the screen and the laptop, especially between the light and dark variants. Toggling dark mode has been my favorite fidget spinner. Creating light/dark variants is a bit tedious to do manually every release, but automating still a bit too hard to pull off (the taking screenshots of a nightly OS bit). There's also the fun of picking a theme for the screenshot rather than doing the same thing over and over. Doing the screenshooting manually meant automating the rest, as a 6 month cycle is enough time to forget how things are done. The process is held together with duct tape, I mean a python script, that renders the website image assets from the few screenshots captured using GNOME OS running inside Boxes. Two great invisible things made by amazing individuals that could go away in an instant and that thought gives me a dose of anxiety. This does take a minute to render on a laptop (CPU only Cycles), but is a matter of a single invocation and a git commit. So far it has survived a couple of Blender releases, so fingers crossed for the future. Sophie has recently been looking into translations, so we might reconsider that 3D approach if translated screenshots become viable (and have them contained in an SVG similar to how os.gnome.org is done). So far the 3D hero has always been in sync with the release, unlike in our Wordpress days. Fingers crossed.

- This Week in GNOME: #226 Exporting Events (2025/11/21 00:00)
  Update on what happened across the GNOME project in the week from November 14 to November 21. GNOME Core Apps and Libraries Calendar ↗ A simple calendar application. Hari Rana | TheEvilSkeleton (any/all) □□ □⚧ says Thanks to FineFindus, who previously worked on exporting events as .ics files, GNOME Calendar can now export calendars as .ics files, courtesy of merge request !615! This will be available in GNOME 50. Hari Rana | TheEvilSkeleton (any/all) □□ □⚧ says After two long and painful years, several design iterations, and more than 50 rebases later, we finally merged the infamous, trauma-inducing merge request !362 on GNOME Calendar. This changes the entire design of the quick-add popover by merging both pages into one and updating the style to conform better with modern GNOME designs. Additionally, it remodels the way the popover retrieves and displays calendars, reducing 120 lines of code. The calendars list in the quick-add popover has undergone accessibility improvements, providing a better experience for assistive technologies and keyboard users. Specifically: tabbing from outside the list will focus the selected calendar in the list; tabbing from inside the list will skip the entire list; arrow keys automatically select the focused calendar; and finally, assistive technologies now inform the user of the checked/selected state. Admittedly, the quick-add popover is currently unreachable via keyboard because we lack the resources to implement keyboard focus for month and week cells. We are currently trying to address this issue in merge request !564, and hope to get it merged for GNOME 50, but it's a significant undertaking for a single unpaid developer. If it is not too much trouble, I would really appreciate some donations, to keep me motivated to improve accessibility throughout GNOME and sustain myself: https://tesk.page/#donate This merge request allowed us to close 4 issues, and will be available in GNOME 50. Files ↗ Providing a simple and integrated way of managing your files and browsing your file system. Peter Eisenmann says Files landed two big changes by Khalid Abu Shawarib this week. The first change adds a bunch of tests, bringing the total coverage of the huge code base close to 30%. This will prevent regressions in

previously uncovered areas such as bookmarking or creating files. The second change is more noticeable as the way thumbnails are loaded was largely rewritten to finally make full use of GTK4's recycling views. It took a lot of code detangling to get thumbnails to load asynchronously, but the result is a great speedup, making thumbnails show as fast as never before. □ Attached is a comparison of reloading a folder before and after the change Libadwaita ↗ Building blocks for modern GNOME apps using GTK4. Alice (she/her) □⚧□□ announces as of today, libadwaita has support for the new reduced motion preference, both supporting the @media (prefers-reduced-motion: reduce) query from CSS, and using simple crossfade transitions where appropriate (e.g. in AdwDialog, AdwNavigationView and AdwTabOverview Alice (she/her) □⚧□□ reports libadwaita has deprecated the style-dark.css, style-hc.css and style-hc-dark.css resources that AdwApplication automatically loads. They still work, but will be removed in 2.0. Applications are recommended to switch to style.css and media queries for dark and high contrast styles GTK ↗ Cross-platform widget toolkit for creating graphical user interfaces. Matthias Clasen reports This weeks GTK 4.21.2 release includes initial support for the CSS backdrop-filter property. The GSK APIs enabling this are new copy/paste and composite render nodes, which allow flexible reuse of the 'background' at any point in the scene graph. We are looking forward to your experiments with this! GLib ↗ The low-level core library that forms the basis for projects such as GTK and GNOME. Philip Withnall says Luca Bacci has dug into an intermittent output buffering issue with GLib on Windows, which should fix some CI issues and opt various GLib utilities into more modern features on Windows — https://gitlab.gnome.org/GNOME/glib/-/merge_requests/4788 Third Party Projects Alain announces Planify 4.16.0 — Natural dates, smoother flows, and smarter task handling This week, Planify released version 4.16.0, bringing several improvements that make task management faster, more intuitive, and more predictable on GNOME. The highlight of this release is natural language date parsing, now enabled by default in Quick Add. You can type things like "tomorrow 3pm", "next Monday", "25/12/2024", or "ahora", and Planify will automatically convert it into a proper scheduled date. Spanish support has also been added, including expressions like mañana, pasado mañana, próxima semana, and more. Keyboard navigation got a boost too: Ctrl + D now opens the date picker instantly Ctrl + K toggles "Keep adding" mode And several shortcuts were cleaned up for more predictable behavior Planify also adds label management in the task context menu, making it easier to add or remove labels without opening the full editor. For calendar users, event items now open a richer details popover, with automatic detection of Google Meet and Microsoft Teams links, making online meetings just one click away. As always, translations, bug fixes, and general UI refinements round out the update. Planify 4.16.0 is available now on Flathub Jan-Willem reports This week I released Java-GI version 0.13.0, a Java language binding for GNOME and other libraries that support GObject-Introspection, based on OpenJDK's new FFM functionality. Some of the highlights in this release are: Bindings for LibRsvg, GstApp (for GStreamer) and LibSecret have been added The website for Java-GI has its own domain name now: java-gi.org, and this is also used in all module- and package names Thanks to GObject-Introspection's extensive testsuite, I've implemented over 900 testcases to test the Java bindings, and fixed many bugs along the way. I hope that Java-GI will help Java (or Kotlin, Scala, Clojure, …) developers to create awesome new GNOME apps! Quadrapassel ↗ Fit falling blocks together. Will Warner says Quadrapassel 49.2 is out! Here is whats new: Updated translations: Ukrainian, Russian, Brazilian Portuguese, Chinese (China), Slovenian, Georgian Made the 'P' key pause the game Replaced the user help docs with a 'Game Rules' dialog Stopped the menu button taking focus Fixed a bug where the game's score would not be recorded when the app was quit Added total rows and level information to scores Phosh ↗ A pure wayland shell for mobile devices. Guido announces Phosh 0.51.0 is out: There's a new quick setting that allows to toggle location services on/off and the ☕ quick setting can now disable itself after a certain amount of time (check here on how to configure the intervals). We also add added a toggle to enable automatic brightness from the top panel and when enabled the brightness slider acts as an offset to the current brightness value. The minimum brightness of the □

brightness slider can now be configured via hwdb/udev allowing one go to lower values then the former hard coded 40%. The configuration is maintained in gmobile. If you're using Phosh on a Google Pixel 3A XL you can now enjoy haptic feedback when typing on the on screen keyboard (like users on other devices) and creating notch configurations for new devices should now be simpler as our tooling can take screen shots of the resulting UI element layout in Phosh for you. There's more, see the full details at here GNOME Websites Emmanuele Bassi says After a long time, the new user help website is now available and up to date with the latest content. The new help website replaces the static snapshot of the old library-web project, but it is still a work in progress, and contributions are welcome. Just like in the past, the content is sourced from each application, as well as from the gnome-user-docs repository. If you want to improve the documentation of GNOME components and core applications, make sure to join the #docs:gnome.org room. Shell Extensions Pedro Sader Azevedo announces Foresight is a GNOME Shell extension that automatically enters the activities view on empty workspaces, making it faster to open apps and start using your computer! This week, it gained support for GNOME 49, courtesy of gabrielpalassi. This is the second time in a row that Foresight gained support for a newer GNOME Shell version thanks to community contributions, which I'm immensely grateful for. I'm also very grateful to Just Perfection, who single-handedly holds so many responsibilities in the GNOME Shell extensions ecosystem. The latest version of Foresight is available at EGO: https://extensions.gnome.org/extension/7901/foresight/ Happy foretelling □□ Miscellaneous revisto reports The Persian GNOME community was featured at the Debian 13 Release Party at Sharif University in Iran. The talk introduced GNOME, explained how the Persian community came together, highlighted its contributions (GTK/libadwaita apps, GNOME Circle involvement, translations, and fa.gnome.org), and invited newcomers to participate and contribute. Recording available (Farsi): https://youtu.be/UPmNNygNQuc GNOME Foundation ramcq reports The GNOME Foundation board has shared details about our recently-approved balanced budget for 2024-25, as well as a note to share our thanks to Karen Sandler, as she has decided to step down from the board. 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!

- Philip Withnall: Parental controls screen time limits backend (2025/11/19 23:39)
  Ignacy blogged recently about all the parts of the user interface for screen time limits in parental controls in GNOME. He's been doing great work pulling that all together, while I have been working on the backend side of things. We're aiming for this screen time limits feature to appear in GNOME 50. High level design There's a design document which is the canonical reference for the design of the backend, but to summarise it at a high level: there's a stateless daemon, malcontent-timerd, which receives logs of the child user's time usage of the computer from gnome-shell in the child's session. For example, when the child stops using the computer, gnome-shell will send the start and end times of the most recent period of usage. The daemon deduplicates/merges and stores them. The parent has set a screen time policy for the child, which says how much time they're allowed on the computer per day (for example, 4h at most; or only allowed to use the computer between 15:00 and 17:00). The policy is stored against the child user in accounts-service. malcontent-timerd applies this policy to the child's usage information to calculate an 'estimated end time' for the child's current session, assuming that they continue to use the computer without taking a break. If they stop or take a break, their usage – and hence the estimated end time – is updated. The child's gnome-shell is notified of changes to the estimated end time and, once it's reached, locks the child's session (with appropriate advance warning). Meanwhile, the parent can query the child's computer usage via a separate API to malcontent-timerd. This returns the child's total screen time usage per day, which allows the usage chart to be shown to the parent in the parental controls user interface (malcontent-control). The daemon imposes access controls on which users can query for usage information. Because the daemon can be accessed by the child and by the parent, and needs to be write-only for the child and read-only for the

parent, it has to be a system daemon. There’s a third API flow which allows the child to request an extension to their screen time for the day, but that’s perhaps a topic for a separate post. IPC diagram of screen time limits support in malcontent. Screen time limit extensions are shown in dashed arrows. So, at its core, malcontent-timerd is a time range store with some policy and a couple of D-Bus interfaces built on top. Per-app time limits Currently it only supports time limits for login sessions, but it is built in such a way that adding support for time limits for specific apps would be straightforward to add to malcontent-timerd in future. The main work required for that would be in gnome-shell — recording usage on a per-app basis (for apps which have limits applied), and enforcing those limits by freezing or blocking access to apps once the time runs out. There are some interesting user experience questions to think about there before anyone can implement it — how do you prevent a user from continuing to use an app without risking data loss (for example, by killing it)? How do you unambiguously remind the user they’re running out of time for a specific app? Can we reliably find all the windows associated with a certain app? Can we reliably instruct apps to save their state when they run out of time, to reduce the risk of data loss? There are a number of bits of architecture we’d need to get in place before per-app limits could happen. Wrapping up As it stands though, the grant funding for parental controls is coming to an end. Ignacy will be continuing to work on the UI for some more weeks, but my time on it is basically up. With the funding, we’ve managed to implement digital wellbeing (screen time limits and break reminders for adults) including a whole UI for it in gnome-control-center and a fairly complex state machine for tracking your usage in gnome-shell; a refreshed UI for parental controls; parental controls screen time limits as described above; the backend for web filtering (but more on that in a future post); and everything is structured so that the extra features we want in future should bolt on nicely. While the features may be simple to describe, the implementation spans four projects, two buses, contains three new system daemons, two new system data stores, and three fairly unique new widgets. It’s tackled all sorts of interesting user design questions (and continues to do so). It’s fully documented, has some unit tests (but not as many as I’d like), and can be integration tested using sysexts. The new widgets are localisable, accessible, and work in dark and light mode. There are even man pages. I’m quite pleased with how it’s all come together. It’s been a team effort from a lot of people! Code, design, input and review (in no particular order): Ignacy, Allan, Sam, Florian, Sebastian, Matthijs, Felipe, Rob. Thank you Endless for the grant and the original work on parental controls. Administratively, thank you to everyone at the GNOME Foundation for handling the grant and paperwork; and thank you to the freedesktop.org admins for providing project hosting for malcontent!

- Lennart Poettering: Mastodon Stories for systemd v258 (2025/11/18 00:00)
  Already on Sep 17 we released systemd v258 into the wild. In the weeks leading up to that release I have posted a series of serieses of posts to Mastodon about key new features in this release, under the #systemd258 hash tag. It was my intention to post a link list here on this blog right after completing that series, but I simply forgot! Hence, in case you aren't using Mastodon, but would like to read up, here's a list of all 37 posts: Post #1: systemctl start -v Post #2: Home areas Post #3: systemd-resolved delegate zones Post #4: Foreign UID range Post #5: /etc/hostname ??? wildcards Post #6: Quota on /tmp/ Post #7: ConcurretnySoftMax= + ConcurrencyHardMax= Post #8: Product UUID in ConditionHost= Post #9: Context OSC terminal sequences Post #10: uki-url Boot Loader Spec Type #1 fields Post #11: rd.break= boot breakpoints Post #12: Factory Reset Rework Post #13: systemd-resolved DNS Configuration Change IPC Subscription API Post #14: io.systemd.boot-entries.extra= SMBIOS Type #11 Key Post #15: Bring Your Own Firmware Post #16: userdb record aliases Post #17: systemd-validatefs and its xattrs Post #18: Offline Signing of Artifacts Post #19: PAMName= in services hooked up to ask-password protocol Post #20: x-systemd.graceful-option= mount option Post #21: systemd-userdb-load-credentials.service Post #22: systemd-vmspawn --grow-image=a Post #23: systemd-notify --fork Post #24: $TERM auto-discovery Post #25: Rebooting/Powering off systemd-nspawn containers via hotkey Post #26: ExecStart= | modifier Post #27:

systemctl reload reloads confexts Post #28: Server side userdb filtering Post #29: Quota on StateDirectory= and friends Post #30: systemd-analyze unit-shell Post #31: /etc/issue.d/ drop-in for AF_VSOCK CID Post #32: fsverity in systemd-repart Post #33: AcceptFileDescriptor= + PassPIDFD= Post #34: Tab completion in interactive systemd-firstboot Post #35: rd.systemd.pull= kernel command line option/Boot into tarball Post #36: ConditionKernelModuleLoaded= Post #37: systemd-analyze chid Post #38: homectl list-signing-keys/get-signing-key/add-signing-key/remove-signing-key Post #39: DDI Image Filters Post #40: Android USB Debugging udev rules Post #41: systemd-vmspawn's --smbios11= switch Post #42: $MAINPIDFDID + $MANAGERPIDFDID Post #43: $DEBUG_INVOCATION=1 Respected by all systemd services Post #44: LoaderDeviceURL EFI Variable and systemd.pull='s origin kernel command line switch Post #45: cgroupv1 removal Post #46: ProtectHostname=private Post #47: homectl adopt + homectl register Post #48: systemd-machined Varlink APIs Post #49: DeferTrigger and "lenient" job mode Post #50: Automatic Removal of foreign UID owned delegate subgroups in the per-user service manager Post #51: Per-user ask-password protocol Post #52: PrivateUsers=full Post #53: LoadCredentialEncrypted= in the per-user service manager Post #54: dissect_image builtin in systemd-udevd Post #55: BPF Delegation via Tokens I intend to do a similar series of serieses of posts for the next systemd release (v259), hence if you haven't left tech Twitter for Mastodon yet, now is the opportunity. We intend to shorten the release cycle a bit for the future, and in fact managed to tag v259-rc1 already yesterday, just 2 months after v258. Hence, my series for v259 will begin soon, under the #systemd259 hash tag. In case you are interested, here is the corresponding blog story for systemd v257, and here for v256.

- Code of Conduct Committee: Transparency report for May 2025 to October 2025 (2025/11/15 18:19)
  GNOME’s Code of Conduct is our community’s shared standard of behavior for participants in GNOME. This is the Code of Conduct Committee’s periodic summary report of its activities from May 2025 to October 2025. The current members of the CoC Committee are: Anisa Kuci Carlos Garnacho Christopher Davis Federico Mena Quintero Michael Downey Rosanna Yuen All the members of the CoC Committee have completed Code of Conduct Incident Response training provided by Otter Tech, and are professionally trained to handle incident reports in GNOME community events. The committee has an email address that can be used to send reports: conduct@gnome.org as well as a website for report submission: https://conduct.gnome.org/ Reports Since May 2025, the committee has received reports on a total of 25 possible incidents. Many of these were not actionable; all the incidents listed here were resolved during the reporting period. Report on a conspiracy theory, closed as not actionable. Report that was not actionable. Report about a blog post; not a CoC violation and not actionable. Report about interactions in GitLab; not a CoC violation and not actionable. Report about a blog post; not a CoC violation and not actionable. Question about an Export Control Classification Number (ECCN) for GDM; redirected to discourse.gnome.org. Report about a reply in GitLab; not a CoC violation; pointed out resources about unpaid/volunteer work in open source. Report about a reply in GitLab; not a CoC violation but using language against the community guidelines; sent a reminder to the reported person to use non-violent communication. Two reports about a GNOME Shell extension; recommended actions to take to the extension reviewers. Report about another GNOME Shell extension; recommended actions to take to the extension reviewers. Multiple reports about a post on planet.gnome.org; removed the post from the feed and its site. Report with a fake attribution; closed as not actionable. Report with threats; closed as not actionable. Report with a fake attribution; closed as not actionable. Report that was not actionable. Support request; advised reporter to direct their question to the infrastructure team. Report closed due to not being actionable; gave the reporter advice on how to deal with their issue. Report about a reply in GitLab; reminded both the reporter and reported person how to communicate appropriately. Report during GUADEC about an incident during the conference; in-person reminder to the reported individual to mind their behavior. Report about a long-standing GitLab interaction; sent a request for a behavior change to the reported

person. Report on a conspiracy theory, closed as not actionable. Report about a Mastodon post, closed as it is not a CoC violation. Report closed due to not being actionable, and not a CoC violation. Report closed due to not being actionable, and not a CoC violation. Report closed due to not being actionable, and not a CoC violation. Meetings of the CoC committee The CoC committee has two meetings each month for general updates, and weekly ad-hoc meetings when they receive reports. There are also in-person meetings during GNOME events. Ways to contact the CoC committee https://conduct.gnome.org – contains the GNOME Code of Conduct and a reporting form. conduct@gnome.org – incident reports, questions, etc.

- Allan Day: GNOME Foundation Update, 2025-11-14 (2025/11/14 18:09)
  This post is another in my series of GNOME Foundation updates, each of which provides an insight into what's happened at the GNOME Foundation over the past week. If you are new to these posts I would encourage you to look over some of the previous entries – there's a fair amount going on at the Foundation right now, and my previous posts provide some useful background. Old business It has been another busy week at the GNOME Foundation. Here's a quick summary: We had a regular Board meeting (as in, the meeting was part of our regular schedule), where we discussed details about the annual report, some financial policy questions, and partnerships. There was another planning meeting for the Digital Wellbeing program, which is close to wrapping up. If you haven't seen it already, Ignacy gave a great overview of the work that's been done on this! There have been more meetings with Dawn Matlak, our new finance advisor and systems guru. We are now at the stage where our new finance system is being setup, which is exciting! The plan is to consolate our payments processing on this new platform, which will reduce operational complexity. Invoice processing in future will also be highly automated, and we are going to get additional capabilities around virtual credit cards, which we already have plans for. Preparations continued for GNOME.Asia 2025, which is happening in Tokyo next month. Assisting attendees with visas and travel has been a particular focus. Most of these items are a continuation of activities that I've described in more detail in previous posts, and I'm a bit light on new news this week, but I think that's to be expected sometimes! Post #10 This is the tenth in my series of GNOME Foundation updates, and this seems like a good point to reflect on how they are going. The weekly posting cadance made sense in the beginning, and wrapping up the week on a Friday afternoon is quite enjoyable, but I am unsure if a weekly post is too much reading for some. So, I'd love to hear feedback: do you like the weekly updates, or do you find it hard to keep up? Would you prefer a higher-level monthly update? Do you like hearing about background operational details, or are you more interested in programs, events and announcements? Answers to these questions would be extremely welcome! Please let me know what you think, either in the comments or by reaching out on Matrix. That's it from me for now. Thanks for reading, and have a great day.
- Gedit Technology blog: Mid-November News (2025/11/14 10:00)
  Misc news about the gedit text editor, mid-November edition! Website: new design Probably the highlight this month is the new design of the gedit website. If it looks familiar to some of you, it's normal, it's because it's an adaptation of the previous GtkSourceView website that was developed in the old gnomeweb-wml repository. gnomeweb-wml (projects.gnome.org) is what predates all the wiki pages for Apps and Projects. The wiki has been retired, so another solution had to be found. For the timeline, projects.gnome.org was available until 2013/2014 where all the content had been migrated to the wiki. Then the wiki has been retired in 2024. Note that there are still rough edges on the gedit website, and more importantly some efforts still need to be done to bring the old CSS stylesheet forward to the new(-ish) responsive web design world. For the most nostalgic of you: gedit website in 2013 (projects.gnome.org/gedit/ on web.archive.org) GtkSourceView website in 2013 (projects.gnome.org/gtksourceview/ on web.archive.org) And for the least nostalgic of you: gedit website from last month (October 2025) What

we can say is that the gedit project has stood the test of time! Enter TeX: improved search and replace Some context: I would like some day to unify the search and replace feature between Enter TeX and gedit. It needs to retain the best of each. In Enter TeX it's a combined horizontal bar, something that I would like in gedit too to replace the dialog window that occludes part of the text. In gedit the strengths include: the search-as-you-type possibility, and a history of past searches. Both are missing in Enter TeX. (These are not the only things that need to be retained; the same workflows, keyboard shortcuts etc. are also an integral part of the functionality). So to work towards that goal, I started in Enter TeX. I merged around 50 commits in the git repository for this change already, rewriting in C (from Vala) some parts and improving the UI along the way. The code needs to be in C because it'll be moved to libgedit-tepl so that it can be consumed by gedit easily. Here is how it looks: Internal refactoring for GeditWindow and its statusbar GeditWindow is what we can call a god class. It is too big, both in the number of lines and the number of instance variables. So this month I've continued to refactor it, to extract a GeditWindowStatus class. There was already a GeditStatusbar class, but its features have now been moved to libgedit-tepl as TeplStatusbar. GeditWindowStatus takes up the responsibility to create the TeplStatusbar, to fill it with the indicators and other buttons, and to make the connection with GeditWindow and the current tab/document. So as a result, GeditWindow is a little less omniscient ;-) As a conclusion gedit does not materialize out of empty space; it takes time to develop and maintain. To demonstrate your appreciation of this piece of software and help its future development, remember that you can fund the project. Your support is critical and much appreciated.

- Andy Wingo: the last couple years in v8's garbage collector (2025/11/13 15:21)
  Let’s talk about memory management! Following up on my article about 5 years of developments in V8’s garbage collector, today I’d like to bring that up to date with what went down in V8’s GC over the last couple years.methodololologyI selected all of the commits to src/heap since my previous roundup. There were 1600 of them, including reverts and relands. I read all of the commit logs, some of the changes, some of the linked bugs, and any design document I could get my hands on. From what I can tell, there have been about 4 FTE from Google over this period, and the commit rate is fairly constant. There are very occasional patches from Igalia, Cloudflare, Intel, and Red Hat, but it’s mostly a Google affair.Then, by the very rigorous process of, um, just writing things down and thinking about it, I see three big stories for V8’s GC over this time, and I’m going to give them to you with some made-up numbers for how much of the effort was spent on them. Firstly, the effort to improve memory safety via the sandbox: this is around 20% of the time. Secondly, the Oilpan odyssey: maybe 40%. Third, preparation for multiple JavaScript and WebAssembly mutator threads: 20%. Then there are a number of lesser side quests: heuristics wrangling (10%!!!!), and a long list of miscellanea. Let’s take a deeper look at each of these in turn.the sandboxThere was a nice blog post in June last year summarizing the sandbox effort: basically, the goal is to prevent user-controlled writes from corrupting memory outside the JavaScript heap. We start from the assumption that the user is somehow able to obtain a write-anywhere primitive, and we work to mitigate the effect of such writes. The most fundamental way is to reduce the range of addressable memory, notably by encoding pointers as 32-bit offsets and then ensuring that no host memory is within the addressable virtual memory that an attacker can write. The sandbox also uses some 40-bit offsets for references to larger objects, with similar guarantees. (Yes, a sandbox really does reserve a terabyte of virtual memory).But there are many, many details. Access to external objects is intermediated via type-checked external pointer tables. Some objects that should never be directly referenced by user code go in a separate “trusted space”, which is outside the sandbox. Then you have read-only spaces, used to allocate data that might be shared between different isolates, you might want multiple cages, there are “shared” variants of the other spaces, for use in shared-memory multi-threading, executable code spaces with embedded object references, and so on and so on. Tweaking, elaborating, and maintaining all of these

details has taken a lot of V8 GC developer time.I think it has paid off, though, because the new development is that V8 has managed to turn on hardware memory protection for the sandbox: sandboxed code is prevented by the hardware from writing memory outside the sandbox.Leaning into the “attacker can write anything in their address space” threat model has led to some funny patches. For example, sometimes code needs to check flags about the page that an object is on, as part of a write barrier. So some GC-managed metadata needs to be in the sandbox. However, the garbage collector itself, which is outside the sandbox, can’t trust that the metadata is valid. We end up having two copies of state in some cases: in the sandbox, for use by sandboxed code, and outside, for use by the collector.The best and most amusing instance of this phenomenon is related to integers. Google’s style guide recommends signed integers by default, so you end up with on-heap data structures with int32_t len and such. But if an attacker overwrites a length with a negative number, there are a couple funny things that can happen. The first is a sign-extending conversion to size_t by run-time code, which can lead to sandbox escapes. The other is mistakenly concluding that an object is small, because its length is less than a limit, because it is unexpectedly negative. Good times!oilpanIt took 10 years for Odysseus to get back from Troy, which is about as long as it has taken for conservative stack scanning to make it from Oilpan into V8 proper. Basically, Oilpan is garbage collection for C++ as used in Blink and Chromium. Sometimes it runs when the stack is empty; then it can be precise. But sometimes it runs when there might be references to GC-managed objects on the stack; in that case it runs conservatively.Last time I described how V8 would like to add support for generational garbage collection to Oilpan, but that for that, you’d need a way to promote objects to the old generation that is compatible with the ambiguous references visited by conservative stack scanning. I thought V8 had a chance at success with their new mark-sweep nursery, but that seems to have turned out to be a lose relative to the copying nursery. They even tried sticky mark-bit generational collection, but it didn’t work out. Oh well; one good thing about Google is that they seem willing to try projects that have uncertain payoff, though I hope that the hackers involved came through their OKR reviews with their mental health intact.Instead, V8 added support for pinning to the Scavenger copying nursery implementation. If a page has incoming ambiguous edges, it will be placed in a kind of quarantine area for a while. I am not sure what the difference is between a quarantined page, which logically belongs to the nursery, and a pinned page from the mark-compact old-space; they seem to require similar treatment. In any case, we seem to have settled into a design that was mostly the same as before, but in which any given page can opt out of evacuation-based collection.What do we get out of all of this? Well, not only can we get generational collection for Oilpan, but also we unlock cheaper, less bug-prone “direct handles” in V8 itself.The funny thing is that I don’t think any of this is shipping yet; or, if it is, it’s only in a Finch trial to a minority of users or something. I am looking forward in interest to seeing a post from upstream V8 folks; whole doctoral theses have been written on this topic, and it would be a delight to see some actual numbers.shared-memory multi-threadingJavaScript implementations have had the luxury of a single-threadedness: with just one mutator, garbage collection is a lot simpler. But this is ending. I don’t know what the state of shared-memory multi-threading is in JS, but in WebAssembly it seems to be moving apace, and Wasm uses the JS GC. Maybe I am overstating the effort here—probably it doesn’t come to 20%—but wiring this up has been a whole thing.I will mention just one patch here that I found to be funny. So with pointer compression, an object’s fields are mostly 32-bit words, with the exception of 64-bit doubles, so we can reduce the alignment on most objects to 4 bytes. V8 has had a bug open forever about alignment of double-holding objects that it mostly ignores via unaligned loads.Thing is, if you have an object visible to multiple threads, and that object might have a 64-bit field, then the field should be 64-bit aligned to prevent tearing during atomic access, which usually means the object should be 64-bit aligned. That is now the case for Wasm structs and arrays in the shared space.side questsRight, we’ve covered what to me are the main stories of V8’s GC over the past couple years. But let me mention a few funny side quests that I saw.the heuristics two-stepThis one I find to be

hilariousad. Tragicomical. Anyway I am amused. So any real GC has a bunch of heuristics: when to promote an object or a page, when to kick off incremental marking, how to use background threads, when to grow the heap, how to choose whether to make a minor or major collection, when to aggressively reduce memory, how much virtual address space can you reasonably reserve, what to do on hard out-of-memory situations, how to account for off-heap mallocated memory, how to compute whether concurrent marking is going to finish in time or if you need to pause… and V8 needs to do this all in all its many configurations, with pointer compression off or on, on desktop, high-end Android, low-end Android, iOS where everything is weird, something called Starboard which is apparently part of Cobalt which is apparently a whole new platform that Youtube uses to show videos on set-top boxes, on machines with different memory models and operating systems with different interfaces, and on and on and on. Simply tuning the system appears to involve a dose of science, a dose of flailing around and trying things, and a whole cauldron of witchcraft. There appears to be one person whose full-time job it is to implement and monitor metrics on V8 memory performance and implement appropriate tweaks. Good grief!mutex mayhemToon Verwaest noticed that V8 was exhibiting many more context switches on MacOS than Safari, and identified V8's use of platform mutexes as the problem. So he rewrote them to use os_unfair_lock on MacOS. Then implemented adaptive locking on all platforms. Then… removed it all and switched to abseil.Personally, I am delighted to see this patch series, I wouldn't have thought that there was juice to squeeze in V8's use of locking. It gives me hope that I will find a place to do the same in one of my projects :)ta-ta, third-party heapIt used to be that MMTk was trying to get a number of production language virtual machines to support abstract APIs so that MMTk could slot in a garbage collector implementation. Though this seems to work with OpenJDK, with V8 I think the churn rate and laser-like focus on the browser use-case makes an interstitial API abstraction a lose. V8 removed it a little more than a year ago.finSo what's next? I don't know; it's been a while since I have been to Munich to drink from the source. That said, shared-memory multithreading and wasm effect handlers will extend the memory management hacker's full employment act indefinitely, not to mention actually landing and shipping conservative stack scanning. There is a lot to be done in non-browser V8 environments, whether in Node or on the edge, but it is admittedly harder to read the future than the past.In any case, it was fun taking this look back, and perhaps I will have the opportunity to do this again in a few years. Until then, happy hacking!

- Jiri Eischmann: How We Streamed OpenAlt on Vhsky.cz (2025/11/13 11:37)
  The blog post was originally published on my Czech blog. When we launched Vhsky.cz a year ago, we did it to provide an alternative to the near-monopoly of YouTube. I believe video distribution is so important today that it's a skill we should maintain ourselves. To be honest, it's bothered me for the past few years that even open-source conferences simply rely on YouTube for streaming talks, without attempting to secure a more open path. We are a community of tech enthusiasts who tinker with everything and take pride in managing things ourselves, yet we just dump our videos onto YouTube, even when we have the tools to handle it internally. Meanwhile, it's common for conferences abroad to manage this themselves. Just look at FOSDEM or Chaos Communication Congress. This is why, from the moment Vhsky.cz launched, my ambition was to broadcast talks from OpenAlt—a conference I care about and help organize. The first small step was uploading videos from previous years. Throughout the year, we experimented with streaming from OpenAlt meetups. We found that it worked, but a single stream isn't quite the stress test needed to prove we could handle broadcasting an entire conference. For several years, Michal Vašíček has been in charge of recording at OpenAlt, and he has managed to create a system where he handles recording from all rooms almost single-handedly (with assistance from session chairs in each room). All credit to him, because other conferences with a similar scope of recordings have entire teams for this. However, I don't have insight into this part of the process, so I won't focus on it. Michal's job was to get the streams to our server; our job was to get them

to the viewers. OpenAlt's AV background with running streams. Author: Michal Stanke. Stress Test We only got to a real stress test the weekend before the conference, when Bashy prepared a setup with seven streams at 1440p resolution. This was exactly what awaited us at OpenAlt. Vhsky.cz runs on a fairly powerful server with a 32-core i9-13900 processor and 96 GB of RAM. However, it's not entirely dedicated to PeerTube. It has to share the server with other OSCloud services (OSCloud is a community hosting of open source web services). We hadn't been limited by performance until then, but seven 1440p streams were truly at the edge of the server's capabilities, and streams occasionally dropped. In reality, this meant 14 continuous transcoding processes, as we were streaming in both 1440p and 480p. Even if you don't change the resolution, you still need to transcode the video to leverage useful distribution features, which I'll cover later. The 480p resolution was intended for mobile devices and slow connections. Remote Runner We knew the Vhsky.cz server alone couldn't handle it. Fortunately, PeerTube allows for the use of "remote runners". The PeerTube instance sends video to these runners for transcoding, while the main instance focuses only on distributing tasks, storage, and video distribution to users. However, it's not possible to do some tasks locally and offload others. If you switch transcoding to remote runners, they must handle all the transcoding. Therefore, we had to find enough performance somewhere to cover everything. I reached out to several hosting providers known to be friendly to open-source activities. Adam Štrauch from Roští.cz replied almost immediately, saying they had a backup machine that they had filed a warranty claim for over the summer and hadn't tested under load yet. I wrote back that if they wanted to see how it behaved under load, now was a great opportunity. And so we made a deal. It was a truly powerful machine: a 48-core Ryzen with 1 TB of RAM. Nothing else was running on it, so we could use all its performance for video transcoding. After installing the runner on it, we passed the stress test. As it turned out, the server with the runner still had a large reserve. For a moment, I toyed with the idea of adding another resolution to transcode the videos into, but then I decided we'd better not tempt fate. The stress test showed us we could keep up with transcoding, but not how it would behave with all the viewers. The performance reserve could come in handy. Load on the runner server during the stress test. Author: Adam Štrauch. Smart Video Distribution Once we solved the transcoding performance, it was time to look at how PeerTube would handle video distribution. Vhsky.cz has a bandwidth of 1 Gbps, which isn't much for such a service. If we served everyone the 1440p stream, we could serve a maximum of 100 viewers. Fortunately, another excellent PeerTube feature helps with this: support for P2P sharing using HLS and WebRTC. Thanks to this, every viewer (unless they are on a mobile device and data) also becomes a peer and shares the stream with others. The more viewers watch the stream, the more they share the video among themselves, and the server load doesn't grow at the same rate. A two-year-old stress test conducted by the PeerTube developers themselves gave us some idea of what Vhsky could handle. They created a farm of 1,000 browsers, simulating 1,000 viewers watching the same stream or VOD. Even though they used a relatively low-performance server (quad-core i7-8700 CPU @ 3.20GHz, slow hard drive, 4 GB RAM, 1 Gbps connection), they managed to serve 1,000 viewers, primarily thanks to data sharing between them. For VOD, this saved up to 98% of the server's bandwidth; for a live stream, it was 75%: If we achieved a similar ratio, then even after subtracting 200 Mbps for overhead (running other services, receiving streams, data exchange with the runner), we could serve over 300 viewers at 1440p and multiples of that at 480p. Considering that OpenAlt had about 160 online viewers in total last year, this was a more than sufficient reserve. Live Operation On Saturday, Michal fired up the streams and started sending video to Vhsky.cz via RTMP. And it worked. The streams ran smoothly and without stuttering. In the end, we had a maximum of tens of online viewers at any one time this year, which posed no problem from a distribution perspective. In practice, the server data download savings were large even with just 5 peers on a single stream and resolution. Our solution, which PeerTube allowed us to flexibly assemble from servers in different data centers, has one disadvantage: it creates some latency. In our case, however, this meant the stream on Vhsky.cz was about 5-10 seconds behind the stream

on YouTube, which I don’t think is a problem. After all, we’re not broadcasting a sports event. Diagram of the streaming solution for OpenAlt. Labels in Czech, but quite self-explanatory. Minor Problems We did, however, run into minor problems and gained experience that one can only get through practice. During Saturday, for example, we found that the stream would occasionally drop from 1440p to 480p, even though the throughput should have been sufficient. This was because the player felt that the delivery of stream chunks was delayed and preemptively switched to a lower resolution. Setting a higher cache increased the stream delay slightly, but it significantly reduced the switching to the lower resolution. Subjectively, even 480p wasn’t a problem. Most of the screen was taken up by the red frame with the OpenAlt logo and the slides. The speaker was only in a small window. The reduced resolution only caused slight blurring of the text on the slides, which I wouldn’t even have noticed as a problem if I wasn’t focusing on it. I could imagine streaming only in 480p if necessary. But it’s clear that expectations regarding resolution are different today, so we stream in 1440p when we can. Over the whole weekend, the stream from one room dropped for about two talks. For some rooms, viewers complained that the stream was too quiet, but that was an input problem. This issue was later fixed in the recordings. When uploading the talks as VOD (Video on Demand), we ran into the fact that PeerTube itself doesn’t support bulk uploads. However, tools exist for this, and we’d like to use them next time to make uploading faster and more convenient. Some videos also uploaded with the wrong orientation, which was likely a problem in their metadata, as PeerTube wasn’t the only player that displayed them that way. YouTube, however, managed to handle it. Re-encoding them solved the problem. On Saturday, to save performance, we also tried transcoding the first finished talk videos on the external runner. For these, a bar is displayed with a message that the video failed to save to external storage, even though it is clearly stored in object storage. In the end we had to reupload them because they were available to watch, but not indexed. A small interlude – my talk about PeerTube at this year’s OpenAlt. Streamed, of course, via PeerTube: Thanks and Support I think that for our very first time doing this, it turned out very well, and I’m glad we showed that the community can stream such a conference using its own resources. I would like to thank everyone who participated. From Michal, who managed to capture footage in seven lecture rooms at once, to Bashy, who helped us with the stress test, to Archos and Schmaker, who did the work on the Vhsky side, and Adam Štrauch, who lent us the machine for the external runner. If you like what we do and appreciate that someone is making OpenAlt streams and talks available on an open platform without ads and tracking, we would be grateful if you supported us with a contribution to one of OSCloud’s accounts, under which Vhsky.cz runs. PeerTube is a great tool that allows us to operate such a service without having Google’s infrastructure, but it doesn’t run for free either.

- Ignacy Kuchciński: Digital Wellbeing Contract: Screen Time Limits (2025/11/10 17:31)
  It’s been four months since my last Digital Wellbeing update. In that previous post I talked about the goals of the Digital Wellbeing project. I also described our progress improving and extending the functionality of the GNOME Parental Controls application, as well as redesigning the application to meet the current design guidelines. Introducing Screen Time Limits Following our work on the Parental Controls app, the next major work item was to implement screen time limits functionality, offering the parents ability to check the child’s screen time usage, set the time limits, and lock the child account outside of a specified curfew. This feature actually spanned across *three* different GNOME projects: Parental Controls: Screen Time page was added to the Parental Controls app, so that parents can view child screen usage an set the time limits, it also includes a detailed bar chart Settings: Wellbeing panel needed to make its own time limits settings impossible to change when the child has parental controls session limits enabled, since they are not in effect in such situation. There’s now also a banner with an explanation that guides to the Parental Controls app Shell: Child session needed to actually lock when the limit was reached Out of all of the three above, the Parental Controls and Shell changes have been already merged, while the Settings integration has been through unwritten review during the bi-weekly

Settings meeting and adjusted to the feedback, so it’s only a matter of time now before it reaches the main branch as well. You can find the screenshots of the added functionalities below, and the reference designs can be find in the app-mockups and os-mockups tickets. Child screen usage When viewing a managed account, a summary of screen time is shown with actions for changing further settings, as well as actions to access additional settings for restrictions and filtering. Child account view with added screen time overview and action for more options The Screen Time view shows an overview of the child’s account’s screen time as well as controls which mirror those of the Settings panel to control screen limits and downtime for the child. Screen Time page with detailed screen time records and time limit controls Settings integration On the Settings side, a child account will see a banner in the Wellbeing panel that lets them know some settings cannot be changed, with a link to the Parental Controls app. Wellbeing panel with a banner informing that limits can only be changed in Parental Controls Screen limits in GNOME Shell We have implemented the locking mechanism in GNOME Shell. When a Screen Time limit is reached, the session locks, so that the child can’t use the computer for the rest of the day. Following is a screen cast of the Shell functionality:
https://blogs.gnome.org/ignapk/files/2025/11/time-limits-demo.webm Preventing children from unlocking has not been implemented yet. However, fortunately, the hardest part was implementing the framework for the rest of the code, so hopefully the easier graphical change will take less to implement and the next update will be much sooner than this one. GNOME OS images You don’t have to take my word for it, especially since one can notice I’ve had to cut the recording at one point (forgot that one can’t switch users in the lock screen :P) – you can check out all of these features in 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 Malcontent changes While all of these user facing changes look cool, none of them would be actually possible without the malcontent backend, which Philip Withnall has been working on. While the daily schedule had already been implemented, the daily limit session limit had to be added, as well as malcontent timer daemon API for Shell to use. There has been many other improvements, web filtering daemon has been added, which I’ll use in the future for implementing Web Filtering page in Parental Controls app. Conclusion Our work for the GNOME Foundation is funded by Endless and Dalio Philanthropies, so kudos to them! I want to thank Florian Müllner for his patience too, during the very educative for me merge request review, and answering to all of my Shell development wonderings. I also want to thank Matthijs Velsink and Felipe Borges for finding time to review the Settings integration. Now that this foundation has been made, we’ll be focusing on finishing the last remaining bit of the session limits support in Shell, which is tweaking the appearance of lock screen when the limit is reached, and implementing the ignore button for extending screen limit, as well as notifications, followed by Web Filtering support in Parental Controls. Until next update!

- Luis Villa: Three LLM-assisted projects (2025/11/10 07:24)
  Some notes on my first serious coding projects in something like 20 years, possibly longer. If you’re curious what these projects mean, more thoughts over on the OpenML.fyi newsletter. TLDR A GitHub contribution graph, showing a lot of activity in the past three weeks after virtually none the rest of the year. News, Fixed The “Fix The News” newsletter is a pillar of my mental health these days, bringing me news that the world is not entirely going to hell in a handbasket. And my 9yo has repeatedly noted that our family news diet is “broken” in exactly the way Fix The News is supposed to fix—hugely negative, hugely US-centric. So I asked Claude to create a “newspaper” version of FTN — a two page pdf of some highlights. It was a hit. So I’ve now been working with Claude Code to create and gradually improve a four-days-a-week “News, Fixed” newspaper. This has been super-fun for the whole family—my wife has made various suggestions over my shoulder, my son devours it every morning, and it’s the first serious coding project I’ve tackled in ages. It is almost entirely strictly personal (it still has hard-coded Duke Basketball

schedules) but nevertheless is public and FOSS. (It is even my first usage of reuse.software—and also of SonarQube Server!) Example newspaper here. No matter how far removed you are from practical coding experience, I cannot recommend enough finding a simple, fun project like this that scratches a human itch in your life, and using the project to experiment with the new code tools. Getting Things Done assistant While working on News, Fixed a friend pointed out Steve Yegge's "beads", which reimagines software issue tracking as an LLM-centric activity — json-centric, tracked in git, etc. At around the same time, I was also pointed at Superpowers—essentially, canned "skills" like "teach the LLM, temporarily, how to brainstorm". The two of these together in my mind screamed "do this for your overwhelmed todo list". I've long practiced various bastardized versions of Getting Things Done, but one of the hangups has been that I'm inconsistent about doing the daily/weekly/nth-ly reviews that good GTD really relies on. I might skip a step, or not look through all my huge "someday-maybe" list, or… any of many reasons one can be tired and human when faced with a wall of text. Also, while there are many tools out there to do GTD, in my experience they either make some of the hardest parts (like the reviews) your problem, or they don't quite fit with how I want to do GTD, or both. Hacking on my own prompts to manage the reviews seems to fit these needs to a T. I currently use Amazing Marvin as my main GTD tool. It is funky and weird and I've stuck with it much longer than any other task tracker I've ever used. So what I've done so far: wrapped the Marvin API to extract json discovered the Marvin API is very flaky, so done some caching and validation written a lot of prompts for the various phases/tasks in GTD. These work to varying degrees and I really want to figure out how to collaborate with others on them, because I suspect that as more tools offer LLM-ish APIs (whoa, todoist!) these prompts are where the real fun and action will be. This is all read-only right now because of limitations in the Marvin API but for various reasons I'm not yet ready to embark on building my own entire UI. So this will do for now. But this code, therefore, is very limited to me. The prompts on the other hand… Note that my emphasis is not on "do tasks", it is on helping me stay on priority. Less "chief of staff", more "executive assistant"—both incredibly valuable when done well, but different roles. This is different from some of the use examples for Yegge's Beads, which really are around agents. Also note: the results have been outstanding. I'm getting more easily into my doing zone, I think largely because I have less anxiety about staring at the Giant Wall of Tasks that defines the life of any high-level IC. And my projects are better organized and todos feel more accurate than they have been in a long time, possibly ever. a note on LLMs and issue/TODO tracking It is worth noting that while LLMs are probabilistic/lossy, so they can't find the "perfect" next TODO to work on, that's OK. Personal TODO and software issue tracking are inherently subjective, probabilistic activities—there is no objectively perfect "next best thing to work on", "most important thing to work on", etc. So the fact that an LLM is only probabilistic in identifying the next task to work on is fine—no human can do substantially better. In fact I'm pretty sure that once an issue list is past a certain point, the LLM is likely to be able to do better— if (and like many things LLM, this is a big if) you can provide it with documented standards explaining how you want to do prioritization. (Literally one of the first things I did at my first job was write standards on how to prioritize bugs—the forerunner of this doc—so I have strong opinions, and experience, here.) Skills for license "concluded" While at a recent Linux Foundation event, I was shocked to realize how many very smart people haven't internalized the skills/prompts/context stuff. It's either "you chat with it" or "you train a model". This is not their fault; it is hard to keep up! Of course this came up most keenly in the context of the age-old problem of "how do I tell what license an open source project is under". In other words, what is the difference between "I have scanned this" and "I have reached the zen state of SPDX's 'concluded' field". So … yes, I've started playing with scripts and prompts on this. It's much less further along than the other two projects above, but I think it could be very fruitful if structured correctly. Some potentially big benefits above and beyond the traditional scanning and/or throw a lawyer at it approaches: reporting: my very strong intuition, admittedly not yet tested, is that plain-English reports on factors below, plus links into repos, will be much easier for lawyers to

use as a starting point than the UIs of traditional license-scanner tools. And I suspect ultimately more powerful as well, since they'll be able to draw on some of the things below. context sensitivity: unlike a regexp, an LLM can likely fairly reliably understand from context some of the big failures of traditional pattern matching like "this code mentions license X but doesn't actually include it". issue analysis and change analysis: unlike traditional approaches, LLMs can look at the change history of key files like README and LICENSE and draw useful context from them. "oh hey README mentioned a license change on Nov. 9, 2025, here's what the change was and let's see if there are any corresponding issues and commit logs that explain this change" is something that an LLM really can do. (Also it can do that with much more patience than any human.) ClearlyDefined offers test data on this, by the way — I'm really looking forward to seeing if this can be made actually reliable or not. (And then we can hook up reuse.software on the backend to actually improve the upstream metadata…) But even then, I may not ever release this. There's a lot of real risks here and I still haven't thought them through enough to be comfortable with them. That's true even though I think the industry has persistently overstated its ability to reach useful conclusions about licensing, since it so persistently insists on doing licensing analysis without ever talking to maintainers. More to come? I'm sure there will be more of these. That said, one of the interesting temptations of this is that it is very hard to say "something is done" because it is so easy to add more. (eg, once my personal homebrew News Fixed is done… why not turn it into a webapp? once my GTD scripts are done… why not port the backend? etc. etc.) So we'll see how that goes.

- Victor Ma: Google Summer of Code final report (2025/11/07 00:00)
  For Google Summer of Code 2025, I worked on GNOME Crosswords. GNOME Crosswords is a project that consists of two apps: Crosswords, a crossword player Crossword Editor, a crossword editor. Links Here are links to everything that I worked on. Merge requests Merge requests related to the word suggestion algorithm: Improve word suggestion algorithm Add word-list-tests-utils.c Refactor clue-matches-tests.c by using a fixture Use better test assert macros Add macro to reduce boilerplate code in clue-matches-tests.c Add a macro to simplify the test_clue_matches calls Add more tests to clue-matches-tests.c Use string parameter in macro function Add performance tests to clue-matches-tests.c Make phase 3 of word_list_find_intersection() optional Improve print functions for WordArray and WordSet Other merge requests: Fix and refactor editor puzzle import Add MIME sniffing to downloader Add support for remaining divided cell types in svg.c Fix intersect sort Fix rebus intersection Use a single suggested words list for Editor Issues submitted Issues I submitted on GitLab. Design documents Word suggestion algorithm Intersection-based word suggestion algorithm Forward-checking word suggestion algorithm AC-3-based word suggestion algorithm Grid helpers Other documents Development: Ideas list Editor roadmap thoughts Crossword Editor architecture notes Naming problems Font testing Word suggestion algorithm: CSP notes Miscellaneous papers Sub-alphabet idea Competitive analysis: Survey of existing crossword editors Survey of printing feature in existing editors Other: Review of docs Blog posts Introducing my GSoC 2025 project Coding begins A strange bug Bugs, bugs, and more bugs! My first design doc It's alive! When is an optimization not optimal? This is a test post Journal I kept a daily journal of the things that I was working on. Project summary I improved GNOME Crossword Editor's word suggestion algorithm, by re-implementing it as a forward-checking algorithm. Previously, our word suggestion algorithm only considered the constraints imposed by the intersection where the cursor is. This resulted in frequent dead-end word suggestions, which led to user frustration. To fix this problem, I re-implemented our word suggestion algorithm to consider the constraints imposed by every intersection in the current slot. This significantly reduces the number of dead-end word suggestions and leads to a better user experience. As part of this project, I also researched the field of constraint satisfaction problems and wrote a report on how we can use the AC-3 algorithm to further improve our word suggestion algorithm in the future. I also performed a competitive analysis of other crossword editors on the market and wrote a detailed report, to help identify missing features and guide future

development. Word suggestion algorithm improvements The goal of any crossword editor software is to make it as easy as possible to create a good crossword puzzle. To that end, all crossword editors have a feature called a word suggestion list. This is a dynamic list of words that fit the current slot. It helps the user find words that fit the slots on their grid. In order to generate the word suggestion list, crossword editors use a word suggestion algorithm. The simplest example of a word suggestion algorithm considers two constraints: The size of the current slot. The letters in the current slot. So for example, if the current slot is C A _ S, then this basic word suggestion algorithm would return all four-letter words that start with CA and end in S—such as CATS or CABS, but not COTS. The problem There is a problem with this basic word suggestion algorithm, however. Consider the following grid: +---+---+---+---+ | | | Z | +---+---+---+---+ | | | E | +---+---+---+---+ | | | R | +---+---+---+---+ | W | O | R | | < current slot +---+---+---+---+ 4-Down begins with ZER, so the only word it can be is ZERO. This constrains the bottom-right cell to the letter O. 4-Across starts with WOR. We know that the bottom-right cell must be O, so that means that 4-Across must be WORO. But WORO is not a word. So, 4-Down and 4-Across are both unfillable, because no letter fits in the bottom-right cell. This means that there are no valid word suggestions for either 4-Across or 4-Down. Now, suppose that the current slot is 4-Across. The basic algorithm only considers the constraints imposed by the current slot, and so it returns all words that match the pattern W O R _—such as WORD and WORM. But none of these word suggestions actually fit in the slot—they all cause 4-Down to become some nonsensical word. The problem is that the basic algorithm only looks at the current slot, 4-Across. It does not also look at other slots, like 4-Down. Because of that, the algorithm doesn't realize that 4-Down causes 4-Across to be unfillable. And so, the algorithm generates incorrect word suggestions. Our word suggestion algorithm Our word suggestion algorithm was a bit more advanced than this basic algorithm. Our algorithm considered two constraints: The constraints imposed by the current slot. The constraints imposed by the intersecting slot where the cursor is. This means that our algorithm could actually handle the problematic grid properly if the cursor is on the bottom-right cell. But not if the cursor is on any other cell of 4-Across: Consequences All this means that our word suggestion algorithm was prone to generating dead-end words—words that seem to fit a slot, but that actually lead to an unfillable grid. In the problematic grid example I gave, this unfillability is immediately obvious. The user fills 4-Across with a word like WORM, and they instantly see that this turns 4-Down into ZERM, a nonsense word. That makes this grid not so bad. The worst cases are the insidious ones, where the fact that a word suggestion leads to an unfillable grid is not obvious at first. This leads to a ton of wasted time and frustration for the user. My solution To fix this problem, I re-implemented our word suggestion algorithm to account for the constraints imposed by all the intersecting slots. Now, our word suggestion algorithm correctly handles the problematic grid example: Our new algorithm doesn't eliminate dead-end words entirely. After all, it only checks the intersecting slots of the current slot—it does not also check the intersecting slots of the intersecting slots, etc. However, the constraints imposed by a slot onto the current slot become weaker, the more intersections-removed it is. Consider: in order for a slot that's two intersections away from the current slot to constrain the current slot, it must first constrain a mutual slot (a slot that intersects both of them) enough for that mutual slot to then constrain the current slot. Compare that to a slot that is only one intersection away from the current slot. All it has to do is be constrained enough that it limits what letters the intersecting cell can be. And so, although my changes do not eliminate dead-end words entirely, they do significantly reduce their prevalence, resulting in a much better user experience. The end This concludes my Google Summer of Code 2025 project! I give my thanks to Jonathan Blandford for his invaluable mentorship and clear communication throughout the past six months. And I thank the GNOME Foundation for its participation in GSoC and commitment to open source.

- Bradley M. Kuhn: Managing Diabetes in Software Freedom (2025/11/06 23:50)
  [The below is a cross-post of an article that I published on my blog at Software Freedom Conservancy.] Our member project representatives and

others who collaborate with SFC on projects know that I've been on part-time medical leave this year. As I recently announced publicly on the Fediverse, I was diagnosed in March 2025 with early-stage Type 2 Diabetes. I had no idea that that the diagnosis would become a software freedom and users' rights endeavor. After the diagnosis, my doctor suggested immediately that I see the diabetes nurse-practitioner specialist in their practice. It took some time get an appointment with him, so I saw him first in mid-April 2025. I walked into the office, sat down, and within minutes the specialist asked me to “take out your phone and install the Freestyle Libre app from Abbott”. This is the first (but, will probably not be the only) time a medical practitioner asked me to install proprietary software as the first step of treatment. The specialist told me that in his experience, even early-stage diabetics like me should use a Continuous Glucose Monitor (CGM). CGM's are an amazing (relatively) recent invention that allows diabetics to sample their blood sugar level constantly. As we software developers and engineers know: great things happen when your diagnostic readout is as low latency as possible. CGMs lower the latency of readouts from 3–4 times a day to every five minutes. For example, diabetics can see what foods are most likely to cause blood sugar spikes for them personally. CGMs put patients on a path to manage this chronic condition well. But, the devices themselves, and the (default) apps that control them are hopelessly proprietary. Fortunately, this was (obviously) not my first time explaining FOSS from first principles. So, I read through the license and terms and conditions of the ironically named “Freestyle Libre” app, and pointed out to the specialist how patient-unfriendly the terms were. For example, Abbott (the manufacturer of my CGM) reserves the right to collect your data (anonymously of course, to “improve the product”). They also require patients to agree that if they take any action to reverse engineer, modify, or otherwise do the normal things our community does with software, the patient must agree that such actions “constitute immediate, irreparable harm to Abbott, its affiliates, and/or its licensors”. I briefly explained to the specialist that I could not possibly agree. I began in real-time (still sitting with the specialist) a search for a FOSS solution. As I was searching, the specialist said: “Oh, I don't use any of it myself, but I think I've heard of this ‘open source’ thing — there is a program called xDrip+ that is for insulin-dependent diabetics that I've heard of and some patients report it is quite good”. While I'm (luckily) very far from insulin-dependency, I eventually found the FOSS Android app called Juggluco (a portmanteau for “Juggle glucose”). I asked the specialist to give me the prescription and I'd try Juggluco to see if it would work. CGM's are very small and their firmware is (by obvious necessity) quite simple. As such, their interfaces are standard. CGM's are activated with Near Field Communication (NFC) — available on even quite old Android devices. The Android device sends a simple integer identifier via NFC that activates the CGM. Once activated — and through the 15-day life of the device — the device responds via Bluetooth with the patient's current glucose reading to any device presenting that integer. Fortunately, I quickly discovered that the FOSS community was already “on this”. The NFC activation worked just fine, even on the recently updated “Freestyle Libre 3+”. After the sixty minute calibration period, I had a continuous readout in Juggluco. CGM's lower latency feedback enables diabetics to have more control of their illness management. one example among many: the patient can see (in real time) what foods most often cause blood sugar spikes for them personally. Diabetes hits everyone differently; data allows everyone to manage their own chronic condition better. My personal story with Juggluco will continue — as I hope (although not until after FOSDEM 2026) to become an upstream contributor to Juggluco. Most importantly, I hope to help the app appear in F-Droid. (I must currently side-load or use Aurora Store to make it work on LineageOS.) Fitting with the history that many projects that interact with proprietary technology must so often live through, Juggluco has faced surreptitious removal from Google's Play Store. Abbott even accused Juggluco of using their proprietary libraries and encryption methods, but the so-called “encryption method” is literally sending an single integer as part of NFC activation. While Abbott backed off, this is another example of why the movement of patients taking control of the technology remains essential. FOSS fits perfectly with this goal. Software freedom gives control of technology to those who actually rely on it — rather than

for-profit medical equipment manufacturers. When I returned to my specialist for a follow-up, we reviewed the data and graphs that I produced with Juggluco. I, of course, have never installed, used, or even agreed to Abbott's licenses and terms, so I have never seen what the Abbott app does. I was thus surprised when I showed my specialist Juggluco's summary graphs. He excitedly told me “this is much better reporting than the Abbott app gives you!”. We all know that sometimes proprietary software has better and more features than the FOSS equivalent, so it's a particularly great success when our community efforts outdoes a wealthy 200 billion-dollar megacorp on software features! Please do watch SFC's site in 2026 for more posts about my ongoing work with Juggluco, and please give generously as an SFC Sustainer to help this and our other work continue in 2026!

- Jordan Petridis: DHH and Omarchy: Midlife crisis (2025/11/06 00:12)
  Couple weeks ago Cloudflare announced it would be sponsoring some Open Source projects. Throwing money at pet projects of random techbros would hardly be news, but there was a certain vibe behind them and the people leading them. In an unexpected turn of events, the millionaire receiving money from the billion-dollar company, thought it would be important to devote a whole blog post to random brokeboy from Athens that had an opinion on the Internet. I was astonished to find the blog post. Now that I moved from normal stalkers to millionaire stalkers, is it a sign that I made it? Have I become such a menace? But more importantly: Who the hell even is this guy? D-H-Who? When I was painting with crayons in a deteriorating kindergarten somewhere in Greece, DHH, David Heinemeier Hansson, was busy with dumping Ruby on Rails in the world and becoming a niche tech celebrity. His street cred for releasing Ruby on Rails would later be replaced by his writing on remote work. Famously authoring “Remote: Office Not Required”, a book based on his own company, 37signals. That cultural cache would go out the window in 2022 when he got in hot water with his own employees after an internal review process concluded that 37signals had been less than stellar when it came to handling race and diversity. Said review process culminated in a clash, where the employees were interested in further exploration of the topic, which DHH responded to them with “You are the person you are complaining about” (meaning: you, pointing out a problem, is the problem). No politics at work This incident lead the two founders of 37signals to the executive decision to forbid any kind of “societal and political discussions” inside the company, which, predictably, lead to a third of the company resigning in protest. This was a massive blow to 37signals. The company was famous for being extremely selective when hiring, as well as affording employees great benefits. Suddenly having a third of the workforce resign over disagreement with management sent a far more powerful message than anything they could have imagined. It would become the starting point for the downwards and radicalizing spiral along with the extended and very public crashout DHH will be going through in the coming years. Starting your own conference so you can never be banned from it Subsequently, DHH was uninvited from keynoting at RailsConf on the account of everyone being grossed out about the handling of the matter and in solidarity with the community members along the employees that quit in protest. That, in turn, would lead to the creation of the Rails Foundation and starting Rails World. A new conference about Rails that 100%-swear-to-god was not just about DHH having his own conference where he can keynote and would never be banned. In the following years DHH would go to explore and express all the spectrum of “down the alt-right pipeline” opinions, like: Woke is so bad that it made me a racist People call me racist, but I found a book telling me how nice I am Fat people are bad and should feel bad DEI is bad DEI is bad part 2: trigger the libs with git branches Cheering for Affirmative action being banned in the USA The 80s were so great cause they were not woke Andrew Tate is a martyr Therapy is so bad, that I shall platform transphobes Therapy is so annoying that I’d rather make an Arch distro instead ADHD is not real Evey boy has ADHD Leave your work problems at therapy, instead of making it the problem of your boss (me) People not wanting kids are bad and Jordan Peterson nailed it Elon Musk and DOGE are awesome (Published 20 days after the Seig Hail) This private school

is making my kids GAY London is full of “non-native Brits” and I am going to praise a race purity rally Words are not violence Words ARE violence, when they are against me Omarchy You either log off a hero, or you see yourself create another linux distribution, and having failed the first part, DHH has been pouring his energy into creating a new project. While letting everyone know how he much prefers that than going to therapy. Thus, Omarchy was born, a set of copy pasted Window Manager and Vim configs turned distro. One of the two projects that Cloudflare will be proudly funding shortly. The only possible option for the compositor would be Hyprland, and even though it’s Wayland (bad!), it’s one of the good-non-woke ones. In a similar tone, the project website would be featuring the tight integration of Omarchy with SuperGrok. Rubygems On a parallel track, the entire Ruby community more or less collapsed in the last two months. Long story short, is that one of the major Ruby Central sponsors, Sidekiq, pulled out the funding after DHH was invited to speak at RailsConf 2025. Shopify, where DHH sits in the boards of directors, was quick to save the day and match the lost funding. Coincidentally an (allegedly) takeover of key parts of the Ruby Infrastructure was carried out by Ruby Central and placed under the control of Shopify in the following weeks. This story is ridiculous, and the entire ruby community is imploding following this. There’s an excellent write-up of the story so far here. In a similar note, and at the same time, we also find DHH drooling over Off-brand Peter Thiel and calling for an Anduril takeover of the Nix community in order to purge all the wokes. On Framework At the same time, Framework had been promoting Omarchy in their social media accounts for a good while. And DHH in turn has been posting about how great Framework hardware is and how the Framework CEO is contributing to his Arch Linux reskin. On October 8th, Framework announced its sponsorhip of the Hyprland project, following 37signal doing the same thing couple weeks earlier. On the same day they made another post promoting Omarchy yet again. This caused a huge backlash and overall PR nightmare, with the apex being a forum thread with over 1700 comments so far. The first reply in forum post, comes from Nirav, Framework’s CEO, with a very questionable choice of words: We support open source software (and hardware), and partner with developers and maintainers across the ecosystem. We deliberately create a big tent, because we want open source software to win. We don’t partner based on individual’s or organization’s beliefs, values, or political stances outside of their alignment with us on increasing the adoption of open source software. I definitely understand that not everyone will agree with taking a big tent approach, but we want to be transparent that bringing in and enabling every organization and community that we can across the Linux ecosystem is a deliberate choice. Mentioning twice a “big tent” as the official policy and response to complains about supporting Fascist and Racist shitheads, is nothing sort of digging a hole for yourself so deep it that it reemerges in another continent. Later on, Nirav would mention that they were finalizing sponsorship of the GNOME Foundation (12k/year) and KDE e.V. (10k/year). In the linked page you can also find a listing of Rails World (DHH’s personal conference) for a one time payment of 24k dollars. There has not been an update since, and at no point have they addressed their support and collaboration with DHH. Can’t lose the money cow and free twitter clout I guess. While I would personally would like to see the donation be rejected, I am not involved with the ongoing discussion on the GNOME Foundation side nor the Foundation itself. What I can say is that myself and others from the GNOME OS team, were involved in initial discussions with Framework, about future collaborations and hardware support. GNOME OS, much like the GNOME Flatpak runtime, is very useful as a reference point in order to identify if a bug, in hardware or software, is distro-specific or not. It’s been a month since the initial debacle with Framework. Regardless of what the GNOME Foundation plans on doing, the GNOME OS team certainly does not feel comfortable in further collaboration given how they have handled the situation so far. It’s sad because the people working there understand the issue, but this does not seem to be a trait shared by the management. A software midlife crisis During all this, DHH decided that his attention must be devoted to get into a mouth-off with a greek kid that called him a Nazi. Since this is not violence (see “Words are not violence” essay), he decided to respond in kind, by calling for violence against me (see “Words are violence”

essay). To anyone who knows a nerd or two over the age of 35, all of the above is unsurprising. This is not some grand heel turn, or some brainwashing that DHH suffered. This is straight up a midlife crisis turned fash speedrun. Here's a dude who barely had any time to confront the world before falling into an infinite money glitch in the form of Ruby on Rails, Jeff Bezos throwing him crazy money, Apple bundling his software as a highlighted feature, becoming a "new work" celebrity and Silicon Valley "Guru". Is it any surprise that such a person later would find the most minuscule kind of opposition as an all-out attack on his self-image? DHH has never had the "best" opinions on a range of things, and they have been dutifully documented by others, but neither have many other developers that are also ignorant of topics outside of software. Being insecure about your hairline and masculine aesthetic to the point of adopting the Charles Manson haircut to cover your balding is one thing. However, it is entirely different to become a drop-shipped version of Elon, tweeting all day and stopping only to write opinion pieces that come off as proving others wrong rather than original thoughts. Case in point: DHH recently wrote about how "men who'd prefer to feel useful over being listened to". The piece is unironically titled "Building competency is better than therapy". It is an insane read, and I'll speculate that it feels as if someone, who DHH can't outright dismiss, suggested he goes to therapy. It's a very "I'll show you off in front of my audience" kind of text. Add to that a three year speedrun decrying the "theocracy of DEI" and the seemingly authoritarian powers of "the wokes", all coincidentally starting after he could not get over his employees disagreeing with him on racial sensitivities. How can someone suggest his workers read Ta-Nehisi Coates's "Between the World and Me" and Michelle Alexander's "The New Jim Crow" in the aftermath of George Floyd's killing and the BLM protests. While a couple of months later writing salivating blogposts after the EDL eugenics rally in England and giving the highest possible praise to Tommy Robinson? Can these people be redeemed? It is certainly not going to help that niche celebrities, like DHH, still hold clout and financial power and are able to spout the worst possible takes without any backlash because of their position. A bunch of Ruby developers recently started a petition to get DHH distanced from the community, and it didn't go far before getting brigaded by the worst people you didn't need to know existed. This of course was amplified to oblivion by DHH and a bunch of sycophants chasing the clout provided by being retweeted by DHH. It would shortly be followed by yet another "I'm never wrong" piece. Is there any chance for these people, who are shielded by their well-paying jobs, their exclusively occupational media diet, and stimuli all happen to reinforce the default world view? I think there is hope, but it demands more voices in tech spaces to speak up about how having empathy for others, or valuing diversity is not some grand conspiracy but rather enrichment to our lives and spaces. This comes hand in hand with firmly shutting down concern trolling and ridiculous "extreme centrist" takes where someone is expected to find common ground with others advocating for their extermination. One could argue that the true spirit of FLOSS, which attracted much of the current midlife crisis developers in the first place, is about diversity and empathy for the varied circumstances and opinions that enriched our space. Conclusion I do not know if his heart is filled with hate or if he is incredibly lost, but it makes little difference since this is his output in the world. David, when you read this I hope it will be a wake-up call. It's not too late, you only need to go offline and let people help you. Stop the pathetic TemuElon speedrun and go take care of your kids. Drop the anti-woke culture wars and pick up a Ta-Nehisi Coates book again. To everyone else: Push back against their vile and misanthropic rhetoric at every turn. Don't let their poisonous roots fester into the ground. There is no place for their hate here. Don't let them find comfort and spew their vomit in any public space. Crush Fascism. Free Palestine .

- Sebastian Wick: Flatpak Happenings (2025/11/04 21:28)
  Yesterday I released Flatpak 1.17.0. It is the first version of the unstable 1.17 series and the first release in 6 months. There are a few things which didn't make it for this release, which is why I'm planning to do another unstable release rather soon, and then a stable release still this

year. Back at LAS this year I talked about the Future of Flatpak and I started with the grim situation the project found itself in: Flatpak was stagnant, the maintainers left the project and PRs didn’t get reviewed. Some good news: things are a bit better now. I have taken over maintenance, Alex Larsson and Owen Taylor managed to set aside enough time to make this happen and Boudhayan Bhattcharya (bbhtt) and Adrian Vovk also got more involved. The backlog has been reduced considerably and new PRs get reviewed in a reasonable time frame. I also listed a number of improvements that we had planned, and we made progress on most of them: It is now possible to define which Flatpak apps shall be pre-installed on a system, and Flatpak will automatically install and uninstall things accordingly. Our friends at Aurora and Bluefin already use this to ship core apps from Flathub on their bootc based systems (shout-out to Jorge Castro). The OCI support in Flatpak has been enhanced to support pre-installing from OCI images and remotes, which will be used in RHEL 10 We merged the backwards-compatible permission system. This allows apps to use new, more restricting permissions, while not breaking compatibility when the app runs on older systems. Specifically access to input devices such as gamepads, and access to the USB portal can now be granted in this way. It will also help us to transition to PipeWire. We have up-to-date docs for libflatpak again Besides the changes directly in Flatpak, there are a lot of other things happening around the wider ecosystem: bbhtt released a new version of flatpak-builder Enhanced License Compliance Tools for Flathub Adrian and I have made plans for a service which allows querying running app instances (systemd-appd). This provides a new way of authenticating Flatpak instances and is a prerequisite for nested sandboxing, PipeWire support, and getting rid of the D-Bus proxy. My previous blog post went into a few more details. Our friends at KDE have started looking into the XDG Intents spec, which will hopefully allow us to implement deep-linking, thumbnailing in Flatpak apps, and other interesting features Adrian made progress on the session save/restore Portal Some rather big refactoring work in the Portals frontend, and GDBus and libdex integration work which will reduce the complexity of asynchronous D-Bus What I have also talked about at my LAS talk is the idea of a Flatpak-Next project. People got excited about this, but I feel like I have to make something very clear: If we redid Flatpak now, it would not be significantly better than the current Flatpak! You could still not do nested sandboxing, you would still need a D-Bus proxy, you would still have a complex permission system, and so on. Those problems require work outside of Flatpak, but have to integrate with Flatpak and Flatpak-Next in the future. Some of the things we will be doing include: Work on the systemd-appd concept Make varlink a feasible alternative to D-Bus D-Bus filtering in the D-Bus daemons Network sandboxing via pasta PipeWire policy for sandboxes New Portals So if you’re excited about Flatpak-Next, help us to improve the Flatpak ecosystem and make Flatpak-Next more feasible!

- Rosanna Yuen: Farewell to these, but not adieu… (2025/11/04 17:44)
  – from Farewell to Malta by Lord Byron Friday was my last day at the GNOME Foundation. I was informed by the Board a couple weeks ago that my position has been eliminated due to budgetary shortfalls. Obviously, I am sad that the Board felt this decision was necessary. That being said, I wanted to write a little note to say goodbye and share some good memories. It has been almost exactly twenty years since I started helping out at the GNOME Foundation. (My history with the GNOME Project is even older; I had code in GNOME 0.13, released in March 1998.) Our first Executive Director had just left, and my husband was Board Treasurer at the time. He inherited a large pile of paperwork and an unhappy IRS. I volunteered to help him figure out how to put the pieces together and get our paperwork in order to get the Foundation back in good standing. After several months of this, the Board offered to pay me to keep it organized. Early on, I used to joke that my title should have been “General Dogsbody” as I often needed to help cover all the little things that needed doing. Over time, my responsibilities within the Foundation grew, but the sentiment remained. I was often responsible for making sure everything that needed doing was done, while putting in many of the processes and procedures Foundation uses to keep running. People often under-estimate how much hard work it is to keep an international non-profit like

the GNOME Foundation going. There is a ton of minutia to be dealt with from ever-changing regulations, requirements, and community needs. Even simple-sounding things like paying people is surprisingly hard the moment it crosses borders. It requires dealing with different payment systems, bank rules, currencies, export regulations, and tax regimes. However, it is a necessary quagmire we have to navigate as it is a crucial tool to further the Foundation's mission. Working a GNOME booth Over time, I have filled a multitude of different roles and positions (and had four different official titles doing so). I am proud of all the things I have done. I have been the assistant to six different Executive Directors helping them onboard as they've started. I've been the bookkeeper, accounts receivable, and accounts payable — keeping our books in order, making sure people are paid, and tracking down funds. I've been Vice Treasurer helping put together our budgets, and created the financial slides for the Treasurer, Board, and AGM. I spent countless nights for almost a decade keeping our accounts updated in GnuCash. And every year for the past nineteen years I was responsible for making sure our taxes are done and 990 filed to keep our non-profit status secure. As someone who has always been deeply entrenched in GNOME's finances, I have always been a responsible steward, looking for ways to spend money more prudently while enforcing budgets. When the Foundation expanded after the Endless Grants, I had to help make the Foundation scale. I have done the jobs of Human Resources, Recruiter, Benefits coordinator, and managed the staff. I made sure the Board, Foundation, and staff are insured, and take their legally required training. I have also had to make sure people and contractors are paid and with all the legal formalities taken care of in all the different countries we operate in , so they only have to concern themselves with supporting GNOME's mission. I have had to be the travel coordinator buying tickets for people (and approving community travel). I have also done the jobs of Project Manager, Project Liaison to all our fiscally sponsored projects and subprojects, Shipping, and Receiving. I have been to countless conferences and tradeshows, giving talks and working booths. I have enjoyed meeting so many users and contributors at these events. I even spent many a weekend at the post-office filling out customs forms and shipping out mouse pads, mugs, and t-shirts to donors (back when we tried to do that in-house.) I tended the Foundation mailbox, logging all the checks we get from our donors and schlepping them to the bank. I have served on five GNOME committees providing stability and continuity as volunteers came and went (Travel, Finance, Engagement, Executive, and Code of Conduct). I was on the team that created GNOME's Code of Conduct, spending countless hours working with community members to help craft the final draft. I am particularly proud of this work, and I believe it has had a positive impact on our community. Over the past year, I have also focused on providing what stability I could to the staff and Foundation, getting us through our second financial review, and started preparing for our first audit planned for next March. This was all while doing my best to hold to GNOME's principles, vision, and commitment to free software. But it is the great people within this community that kept me loyally working with y'all year after year, and the appreciation of the amazing project y'all create that matters. I am grateful to the many community members who volunteer their time so selflessly through the years. Old-timers like Sri and Federico that have been on this journey with me since the very beginning. Other folks that I met through the years like Matthias, Christian, Meg, PTomato, and German. And Marina, who we all still miss. So many newcomers that add enthusiasm into the community like Deepesha, Michael, and Aaditya. So many Board members. There have been so many more names I could mention that I apologize if your name isn't listed. Please know that I am grateful for what everyone has brought into the community. I have truly been blessed to know you all. I am also grateful for the folks on staff that have made GNOME such a wonderful place to work through the years. Our former Executive Directors Stormy, Karen, Neil, Holly, and Richard, all of whom have taught me so much. Other staff members that have come and gone through the years, such as Andrea (who is still volunteering), Molly, Caroline, Emmanuele, and Melissa. And, of course, the current staff of Anisa, Bart, and Kristi, in whose hands I know the Foundation will keep thriving. As I said, my job has always been to make sure things go as smoothly as possible. In my mind, what I do

should quiet any waves so that the waves the Foundation makes go into providing the best programming we can — which is why a moment from GUADEC 2015 still pops up in my head. Picture this: we are all in Gothenburg, Sweden, in line registering for GUADEC. We start chatting in line as it was long. I introduce myself to the person behind me and he sputters, “Oh! You’re important!” That threw me for a loop. I had never seen myself that way. My intention has always been to make things work seamlessly for our community members behind the scenes, but it was always extremely gratifying to hear from folks who have been touched by my efforts. GNOME things still to be transferred to the Board. Suitcase in front is full of items for staffing a GNOME Booth. What’s next for me? I have not had the time to figure this out yet as I have been spending my time transferring what I can to the Board. First things first; I need to figure out how to write a resumé again. I would love to continue working in the nonprofit space, and obviously have a love of free software. But I am open to exploring new ideas. If anyone has any thoughts or opportunities, I would love to hear them! This is not adieu; my heart will always be with GNOME. I still have my seat on the Code of Conduct committee and, while I plan on taking a month or so away to figure things out, do plan on returning to do my bit in keeping GNOME a safe place. If you’d like to drop me a line, I’d love to hear from you. Unfortunately the Board has to keep my current GNOME email address for a few months for the transfer, but I can be reached at <rosanna at gnome> for my personal mail. (Thanks, Bart!) Best of luck to the Foundation.

- Felipe Borges: Our Goal with Google Summer of Code: Contributor Selection (2025/11/04 12:54)
  Last week, as I was writing my trip report about the Google Summer of Code Mentor Summit, I found myself going on a tangent about the program in our community, so I decided to split the content off into a couple of posts. In this post, I want to elaborate a bit on our goal with the program and how intern selection helps us with that. I have long been saying that GSoC is not a “pay-for-code” program for GNOME. It is an opportunity to bring new contributors to our community, improve our projects, and sustain our development model. Mentoring is hard and time consuming. GNOME Developers heroically dedicate hours of their weeks to helping new people learn how to contribute. Our goal with GSoC is to attract contributors that want to become GNOME Developers. We want contributors that will spend time helping others learn and keep the torch going. Merge-requests are very important, but so are the abilities to articulate ideas, hold healthy discussions, and build consensus among other contributors. For years, the project proposal was the main deciding factor for a contributor to get an internship with GNOME. That isn’t working anymore, especially in an era of AI-generated proposals. We need to up our game and dig deeper to find the right contributors. This might even mean asking for fewer internship slots. I believe that if we select a smaller group of people with the right motivations, we can give them the focused attention and support to continue their involvement long after the internship is completed. My suggestion for improving the intern selection process is to focus on three factors: History of Contributions in gitlab.gnome.org: applicants should solve a few ~Newcomers issues, report bugs, and/or participate in discussions. This gives us an idea of how they perform in the contributing process as a whole. Project Proposal: a document describing the project’s goals and detailing how the contributors plans to tackle the project. Containing some reasonable time estimates. An interview: a 10 or 15 minutes call where admins and mentors can ask applicants a few questions about their Project Proposal and their History of Contributions. The final decision to select an intern should be a consideration of how the applicant performed across these aspects. Contributor selection is super important, and we must continue improving our process. This is about investing in the long-term health and sustainability of our project by finding and nurturing its future developers. If you want to find more about GSoC with GNOME, visit gsoc.gnome.org
- Andy Wingo: wastrel, a profligate implementation of webassembly (2025/10/30 22:19)
  Hey hey hey good evening! Tonight a quick note on wastrel, a new WebAssembly implementation.a wasm-to-native compiler that goes through

cWastrel compiles Wasm modules to standalone binaries. It does so by emitting C and then compiling that C.Compiling Wasm to C isn't new: Ben Smith wrote wasm2c back in the day and these days most people in this space use Bastien Müller's w2c2. These are great projects! Wastrel has two or three minor differences from these projects. Let's lead with the most important one, despite the fact that it's as yet vaporware: Wastrel aims to support automatic memory managment via WasmGC, by embedding the Whippet garbage collection library. (For the wingolog faithful, you can think of Wastrel as a Whiffle for Wasm.) This is the whole point! But let's come back to it.The other differences are minor. Firstly, the CLI is more like wasmtime: instead of privileging the production of C, which you then incorporate into your project, Wastrel also compiles the C (by default), and even runs it, like wasmtime run.Unlike wasm2c (but like w2c2), Wastrel implements WASI. Specifically, WASI 0.1, sometimes known as "WASI preview 1". It's nice to be able to take the wasi-sdk's C compiler, compile your program to a binary that uses WASI imports, and then run it directly.In a past life, I once took a week-long sailing course on a 12-meter yacht. One thing that comes back to me often is the way the instructor would insist on taking in the bumpers immediately as we left port, that to sail with them was no muy marinero, not very seamanlike. Well one thing about Wastrel is that it emits nice C: nice in the sense that it avoids many useless temporaries. It does so with a lightweight effects analysis, in which as temporaries are produced, they record which bits of the world they depend on, in a coarse way: one bit for the contents of all global state (memories, tables, globals), and one bit for each local. When compiling an operation that writes to state, we flush all temporaries that read from that state (but only that state). It's a small thing, and I am sure it has very little or zero impact after SROA turns locals into SSA values, but we are vessels of the divine, and it is important for vessels to be C worthy.Finally, w2c2 at least is built in such a way that you can instantiate a module multiple times. Wastrel doesn't do that: the Wasm instance is statically allocated, once. It's a restriction, but that's the use case I'm going for.on performanceOh buddy, who knows?!? What is real anyway? I would love to have proper perf tests, but in the meantime, I compiled coremark using my GCC on x86-64 (-02, no other options), then also compiled it with the current wasi-sdk and then ran with w2c2, wastrel, and wasmtime. I am well aware of the many pitfalls of benchmarking, and so I should not say anything because it is irresponsible to make conclusions from useless microbenchmarks. However, we're all friends here, and I am a dude with hubris who also believes blogs are better out than in, and so I will give some small indications. Please obtain your own salt.So on coremark, Wastrel is some 2-5% percent slower than native, and w2c2 is some 2-5% slower than that. Wasmtime is 30-40% slower than GCC. Voilà.My conclusion is, Wastrel provides state-of-the-art performance. Like w2c2. It's no wonder, these are simple translators that use industrial compilers underneath. But it's neat to see that performance is close to native.on wasiOK this is going to sound incredibly arrogant but here it is: writing Wastrel was easy. I have worked on Wasm for a while, and on Firefox's baseline compiler, and Wastrel is kinda like a baseline compiler in shape: it just has to avoid emitting boneheaded code, and can leave the serious work to someone else (Ion in the case of Firefox, GCC in the case of Wastrel). I just had to use the Wasm libraries I already had and make it emit some C for each instruction. It took 2 days.WASI, though, took two and a half weeks of agony. Three reasons: One, you can be sloppy when implementing just wasm, but when you do WASI you have to implement an ABI using sticks and glue, but you have no glue, it's all just i32. Truly excruciating, it makes you doubt everything, and I had to refactor Wastrel to use C's meager type system to the max. (Basically, structs-as-values to avoid type confusion, but via inline functions to avoid overhead.)Two, WASI is not huge but not tiny either. Implementing poll_oneoff is annoying. And so on. Wastrel's WASI implementation is thin but it's still a couple thousand lines of code.Three, WASI is underspecified, and in practice what is "conforming" is a function of what the Rust and C toolchains produce. I used wasi-testsuite to burn down most of the issues, but it was a slog. I neglected email and important things but now things pass so it was worth it maybe? Maybe?on wasi's filesystem sandboxingWASI preview 1 has this "rights" interface that associated capabilities with file descriptors. I think it was

an attempt at replacing and expanding file permissions with a capabilities-oriented security approach to sandboxing, but it was only a veneer. In practice most WASI implementations effectively implement the sandbox via a permissions layer: for example the process has capabilities to access the parents of preopened directories via .., but the WASI implementation has to actively prevent this capability from leaking to the compiled module via run-time checks.Wastrel takes a different approach, which is to use Linux's filesystem namespaces to build a tree in which only the exposed files are accessible. No run-time checks are necessary; the system is secure by construction. He says. It's very hard to be categorical in this domain but a true capabilities-based approach is the only way I can have any confidence in the results, and that's what I did.The upshot is that Wastrel is only for Linux. And honestly, if you are on MacOS or Windows, what are you doing with your life? I get that it's important to meet users where they are but it's just gross to build on a corporate-controlled platform.The current versions of WASI keep a vestigial capabilities-based API, but given that the goal is to compile POSIX programs, I would prefer if wasi-filesystem leaned into the approach of WASI just having access to a filesystem instead of a small set of descriptors plus scoped openat, linkat, and so on APIs. The security properties would be the same, except with fewer bug possibilities and with a more conventional interface.on wtfSo Wastrel is Wasm to native via C, but with an as-yet-unbuilt GC aim. Why?This is hard to explain and I am still workshopping it.Firstly I am annoyed at the WASI working group's focus on shared-nothing architectures as a principle of composition. Yes, it works, but garbage collection also works; we could be building different, simpler systems if we leaned in to a more capable virtual machine. Many of the problems that WASI is currently addressing are ownership-related, and would be comprehensively avoided with automatic memory management. Nobody is really pushing for GC in this space and I would like for people to be able to build out counterfactuals to the shared-nothing orthodoxy.Secondly there are quite a number of languages that are targetting WasmGC these days, and it would be nice for them to have a good run-time outside the browser. I know that Wasmtime is working on GC, but it needs competition :)Finally, and selfishly, I have a GC library! I would love to spend more time on it. One way that can happen is for it to prove itself useful, and maybe a Wasm implementation is a way to do that. Could Wastrel on wasm_of_ocaml output beat ocamlopt? I don't know but it would be worth it to find out! And I would love to get Guile programs compiled to native, and perhaps with Hoot and Whippet and Wastrel that is a possibility.Welp, there we go, blog out, dude to bed. Hack at y'all later and wonderful wasming to you all!

- Thibault Martin: From VS Code to Helix (2025/10/29 12:00)
  I created the website you're reading with VS Code. Behind the scenes I use Astro, a static site generator that gets out of the way while providing nice conveniences. Using VS Code was a no-brainer: everyone in the industry seems to at least be familiar with it, every project can be opened with it, and most projects can get enhancements and syntactic helpers in a few clicks. In short: VS Code is free, easy to use, and widely adopted. A Rustacean colleague kept singing Helix's praises. I discarded it because he's much smarter than I am, and I only ever use vim when I need to fiddle with files on a server. I like when things "Just Work" and didn't want to bother learning how to use Helix nor how to configure it. Today it has become my daily driver. Why did I change my mind? What was preventing me from using it before? And how difficult was it to get there? Automation is a double-edged sword Automation and technology make work easier, this is why we produce technology in the first place. But it also means you grow more dependent on the tech you use. If the tech is produced transparently by an international team or a team you trust, it's fine. But if it's produced by a single large entity that can screw you over, it's dangerous. VS Code might be open source, but in practice it's produced by Microsoft. Microsoft has a problematic relationship to consent and is shoving AI products down everyone's throat. I'd rather use tools that respect me and my decisions, and I'd rather not get my tools produced by already monopolistic organizations. Microsoft is also based in the USA, and the political climate over there makes me want to depend as little as possible on American tools. I know that's a long, uphill battle, but

we have to start somewhere. I'm not advocating for a ban against American tech in general, but for more balance in our supply chain. I'm also not advocating for European tech either: I'd rather get open source tools from international teams competing in a race to the top, rather than from teams in a single jurisdiction. What is happening in the USA could happen in Europe too. Why I feared using Helix I've never found vim particularly pleasant to use but it's everywhere, so I figured I might just get used to it. But one of the things I never liked about vim is the number of moving pieces. By default, vim and neovim are very bare bones. They can be extended and completely modified with plugins, but I really don't like the idea of having extremely customize tools. I'd rather have the same editor as everyone else, with a few knobs for minor preferences. I am subject to choice paralysis, so making me configure an editor before I've even started editing is the best way to tank my productivity. When my colleague told me about Helix, two things struck me as improvements over vim. Helix's philosophy is that everything should work out of the box. There are a few configs and themes, but everything should work similarly from one Helix to another. All the language-specific logic is handled in Language Servers that implement the Language Server Protocol standard. In Helix, first you select text, and then you perform operations onto it. So you can visually tell what is going to be changed before you apply the change. It fits my mental model much better. But there are major drawbacks to Helix too: After decades of vim, I was scared to re-learn everything. In practice this wasn't a problem at all because of the very visual way Helix works. VS Code "Just Works", and Helix sounded like more work than the few clicks from VS Code's extension store. This is true, but not as bad as I had anticipated. After a single week of usage, Helix was already very comfortable to navigate. After a few weeks, most of the wrinkles have been ironed out and I use it as my primary editor. So how did I overcome those fears? What Helped Just Do It I tried Helix. It can sound silly, but the very first step to get into Helix was not to overthink it. I just installed it on my mac with brew install helix and gave it a go. I was not too familiar with it, so I looked up the official documentation and noticed there was a tutorial. This tutorial alone is what convinced me to try harder. It's an interactive and well written way to learn how to move and perform basic operations in Helix. I quickly learned how to move around, select things, surround them with braces or parenthesis. I could see what I was about to do before doing it. This has been epiphany. Helix just worked the way I wanted. Better: I could get things done faster than in VS Code after a few minutes of learning. Being a lazy person, I never bothered looking up VS Code shortcuts. Because the learning curve for Helix is slightly steeper, you have to learn those shortcuts that make moving around feel so easy. Not only did I quickly get used to Helix key bindings: my vim muscle-memory didn't get in the way at all! Better docs The built-in tutorial is a very pragmatic way to get started. You get results fast, you learn hands on, and it's not that long. But if you want to go further, you have to look for docs. Helix has officials docs. They seem to be fairly complete, but they're also impenetrable as a new user. They focus on what the editor supports and not on what I will want to do with it. After a bit of browsing online, I've stumbled upon this third-party documentation website. The domain didn't inspire me a lot of confidence, but the docs are really good. They are clearly laid out, use-case oriented, and they make the most of Astro Starlight to provide a great reading experience. The author tried to upstream these docs, but that won't happen. It looks like they are upstreaming their docs to the current website. I hope this will improve the quality of upstream docs eventually. After learning the basics and finding my way through the docs, it was time to ensure Helix was set up to help me where I needed it most. Getting the most of Markdown and Astro in Helix In my free time, I mostly use my editor for three things: Write notes in markdown Tweak my website with Astro Edit yaml to faff around my Kubernetes cluster Helix is a "stupid" text editor. It doesn't know much about what you're typing. But it supports Language Servers that implement the Language Server Protocol. Language Servers understand the document you're editing. They explain to Helix what you're editing, whether you're in a TypeScript function, typing a markdown link, etc. With that information, Helix and the Language Server can provide code completion hints, errors & warnings, and easier navigation in your code. In addition to Language

Servers, Helix also supports plugging code formatters. Those are pieces of software that will read the document and ensure that it is consistently formatted. It will check that all indentations use spaces and not tabs, that there is a consistent number of space when indenting, that brackets are on the same line as the function, etc. In short: it will make the code pretty. Markdown Markdown is not really a programming language, so it might seem surprising to configure a Language Server for it. But if you remember what we said earlier, Language Servers can provide code completion, which is useful when creating links for example. Marksman does exactly that! Since Helix is pre-configured to use marksman for markdown files we only need to install marksman and make sure it's in our PATH. Installing it with homebrew is enough. $ brew install marksman We can check that Helix is happy with it with the following command $ hx --health markdown Configured language servers: ✓ marksman: /opt/homebrew/bin/marksman Configured debug adapter: None Configured formatter: None Tree-sitter parser: ✓ Highlight queries: ✓ Textobject queries: ✘ Indent queries: ✘ But Language Servers can also help Helix display errors and warnings, and "code suggestions" to help fix the issues. It means Language Servers are a perfect fit for... grammar checkers! Several grammar checkers exist. The most notable are: LTEX+, the Language Server used by Language Tool. It supports several languages but is quite resource hungry. Harper, a grammar checker Language Server developed by Automattic, the people behind WordPress, Tumblr, WooCommerce, Beeper and more. Harper only support English and its variants, but they intend to support more languages in the future. I mostly write in English and want to keep a minimalistic setup. Automattic is well funded, and I'm confident they will keep working on Harper to improve it. Since grammar checker LSPs can easily be changed, I've decided to go with Harper for now. To install it, homebrew does the job as always: $ brew install harper Then I edited my ~/.config/helix/languages.toml to add Harper as a secondary Language Server in addition to marksman [language-server.harper-ls] command = "harper-ls" args = ["--stdio"] [[language]] name = "markdown" language-servers = ["marksman", "harper-ls"] Finally I can add a markdown linter to ensure my markdown is formatted properly. Several options exist, and markdownlint is one of the most popular. My colleagues recommended the new kid on the block, a Blazing Fast equivalent: rumdl. Installing rumdl was pretty simple on my mac. I only had to add the repository of the maintainer, and install rumdl from it. $ brew tap rvben/rumdl $ brew install rumdl After that I added a new language-server to my ~/.config/helix/languages.toml and added it to the language servers to use for the markdown language. [language-server.rumdl] command = "rumdl" args = ["server"] [...] [[language]] name = "markdown" language-servers = ["marksman", "harper-ls", "rumdl"] soft-wrap.enable = true text-width = 80 soft-wrap.wrap-at-text-width = true Since my website already contained a .markdownlint.yaml I could import it to the rumdl format with $ rumdl import .markdownlint.yaml Converted markdownlint config from '.markdownlint.yaml' to '.rumdl.toml' You can now use: rumdl check --config .rumdl.toml . You might have noticed that I've added a little quality of life improvement: soft-wrap at 80 characters. Now if you add this to your own config.toml you will notice that the text is completely left aligned. This is not a problem on small screens, but it rapidly gets annoying on wider screens. Helix doesn't support centering the editor. There is a PR tackling the problem but it has been stale for most of the year. The maintainers are overwhelmed by the number of PRs making it their way, and it's not clear if or when this PR will be merged. In the meantime, a workaround exists, with a few caveats. It is possible to add spaces to the left gutter (the column with the line numbers) so it pushes the content towards the center of the screen. To figure out how many spaces are needed, you need to get your terminal width with stty $ stty size 82 243 In my case, when in full screen, my terminal is 243 characters wide. I need to remove the content column with from it, and divide everything by 2 to get the space needed on each side. In my case for a 243 character wide terminal with a text width of 80 characters: (243 - 80) / 2 = 81 As is, I would add 203 spaces to my left gutter to push the rest of the gutter and the content to the right. But the gutter itself has a width of 4 characters, that I need to remove from the total. So I need to subtract them from the total, which leaves me with 76 characters to add. I can open my

~/.config/helix/config.toml to add a new key binding that will automatically add or remove those spaces from the left gutter when needed, to shift the content towards the center. [keys.normal.space.t] z = ":toggle gutters.line-numbers.min-width 76 3" Now when in normal mode, pressing <kbd>Space</kbd> then <kbd>t</kbd> then <kbd>z</kbd> will add/remove the spaces. Of course this workaround only works when the terminal runs in full screen mode. Astro Astro works like a charm in VS Code. The team behind it provides a Language Server and a TypeScript plugin to enable code completion and syntax highlighting. I only had to install those globally with $ pnpm install -g @astrojs/language-server typescript @astrojs/ts-plugin Now we need to add a few lines to our ~/.config/helix/languages.toml to tell it how to use the language server [language-server.astro-ls] command = "astro-ls" args = ["--stdio"] config = { typescript = { tsdk = "/Users/thibaultmartin/Library/pnpm/global/5/node_modules/typescript/lib" }} [[language]] name = "astro" scope = "source.astro" injection-regex = "astro" file-types = ["astro"] language-servers = ["astro-ls"] We can check that the Astro Language Server can be used by helix with $ hx --health astro Configured language servers: ✓ astro-ls: /Users/thibaultmartin/Library/pnpm/astro-ls Configured debug adapter: None Configured formatter: None Tree-sitter parser: ✓ Highlight queries: ✓ Textobject queries: ✘ Indent queries: ✘ I also like to get a formatter to automatically make my code consistent and pretty for me when I save a file. One of the most popular code formaters out there is Prettier. I've decided to go with the fast and easy formatter dprint instead. I installed it with $ brew install dprint Then in the projects I want to use dprint in, I do $ dprint init I might edit the dprint.json file to my liking. Finally, I configure Helix to use dprint globally for all Astro projects by appending a few lines in my ~/.config/helix/languages.toml. [[language]] name = "astro" scope = "source.astro" injection-regex = "astro" file-types = ["astro"] language-servers = ["astro-ls"] formatter = { command = "dprint", args = ["fmt", "--stdin", "astro"]} auto-format = true One final check, and I can see that Helix is ready to use the formatter as well $ hx --health astro Configured language servers: ✓ astro-ls: /Users/thibaultmartin/Library/pnpm/astro-ls Configured debug adapter: None Configured formatter: ✓ /opt/homebrew/bin/dprint Tree-sitter parser: ✓ Highlight queries: ✓ Textobject queries: ✘ Indent queries: ✘ YAML For yaml, it's simple and straightforward: Helix is preconfigured to use yaml-language-server as soon as it's in the PATH. I just need to install it with $ brew install yaml-language-server Is it worth it? Helix really grew on me. I find it particularly easy and fast to edit code with it. It takes a tiny bit more work to get the language support than it does in VS Code, but it's nothing insurmountable. There is a slightly steeper learning curve than for VS Code, but I consider it to be a good thing. It forced me to learn how to move around and edit efficiently, because there is no way to do it inefficiently. Helix remains intuitive once you've learned the basics. I am a GNOME enthusiast, and I adhere to the same principles: I like when my apps work out of the box, and when I have little to do to configure them. This is a strong stance that often attracts a vocal opposition. I like products that follow those principles better than those who don't. With that said, Helix sometimes feels like it is maintained by one or two people who have a strong vision, but who struggle to onboard more maintainers. As of writing, Helix has more than 350 PRs open. Quite a few bring interesting features, but the maintainers don't have enough time to review them. Those 350 PRs mean there is a lot of energy and goodwill around the project. People are willing to contribute. Right now, all that energy is gated, resulting in frustration both from the contributors who feel like they're working in the void, and the maintainers who feel like there at the receiving end of a fire hose. A solution to make everyone happier without sacrificing the quality of the project would be to work on a Contributor Ladder. CHAOSS' Dr Dawn Foster published a blog post about it, listing interesting resources at the end.

- Jakub Steiner: USB MIDI Controllers on the M8 (2025/10/28 11:04)
  The M8 has extensive USB audio and MIDI capabilities, but it cannot be a USB MIDI host. So you can control other devices through USB MIDI, but cannot sent to it over USB. Control Surface & Pots for M8 Controlling things via USB devices has to be done through the old TRS (A) jacks. There’s

two devices that can aid in that. I've used the RK06 which is very featureful, but in a very clumsy plastic case and USB micro cable that has a splitter for the HOST part and USB Power in. It also sometimes doesn't reset properly when having multiple USB devices attached through a hub. The last bit is why I even bother with this setup. The Dirtywave M8 has amazing support for the Novation Launchpad Pro MK3. Majority of peolpe hook it up directly to the M8 using the TRS MIDI cables. The Launchpad lacks any sort of pots or encoders though. Thus the need to fuss with USB dongles. What you need is to use the Launchpad Pro as a USB controller and shun at the reliable MIDI connection. The RK06 allows to combine multiple USB devices attached through an unpowered USB hub. Because I am flabbergasted how I did things here's a schema that works. If it doesn't work, unplug the RK06 and turn LPPro off and on in the M8. I hate this setup but it is the only compact one that works (after some fiddling that you absolutely hate when doing a gig). Intech Knot The Hungarians behind the Grid USB controlles (with first class Linux support) have a USB>MIDI device called Knot. It has one great feature of a switch between TRS A/B for the non-standard devices. It is way less fiddly than the RK06, uses nice aluminium housing and is sturdier. Hoewer it doesn't seem to work with the Launchpad Pro via USB and it seems to be completely confused by a USB hub, so it's not useful for my use case of multiple USB controllers. Non-compact but Reliable Novation came out with the Launch Control XL, which sadly replaced pots in the old one with encoders (absolute vs relative movement), but added midi in/ou/through with a MIDI mixer even. That way you can avoid USB altogether and get a reliable setup with control surfaces and encoders and sliders. One day someone comes up with a compact midi capable pots to play along with Launchpad Pro ;) This post has been brought to you by an old man who forgets things.

- Colin Walters: Thoughts on agentic AI coding as of Oct 2025 (2025/10/27 21:08)
  Sandboxed, reviewed parallel agents make sense For coding and software engineering, I've used and experimented with various frontends (FOSS and proprietary) to multiple foundation models (mostly proprietary) trying to keep up with the state of the art. I've come to strongly believe in a few things: Agentic AI for coding needs strongly sandboxed, reproducible environments It makes sense to run multiple agents at once AI output definitely needs human review Why human review is necessary Prompt injection is a serious risk at scale All AI is at risk of prompt injection to some degree, but it's particularly dangerous with agentic coding. All the state of the art today knows how to do is mitigate it at best. I don't think it's a reason to avoid AI, but it's one of the top reasons to use AI thoughtfully and carefully for products that have any level of criticality. OpenAI's Codex documentation has a simple and good example of this. Disabling the tests and claiming success Beyond that, I've experienced multiple times different models happily disabling the tests or adding a println!("TODO add testing here") and claim success. At least this one is easier to mitigate with a second agent doing code review before it gets to human review. Sandboxing The "can I do X" prompting model that various interfaces default to is seriously flawed. Anthropic has a recent blog post on Claude Code changes in this area. My take here is that sandboxing is only part of the problem; the other part is ensuring the agent has a reproducible environment, and especially one that can be run in IaaS environments. I think devcontainers are a good fit. I don't agree with the statement from Anthropic's blog without the overhead of spinning up and managing a container. I don't think this is overhead for most projects because Where it feels like it has overhead, we should be working to mitigate it. Running code as separate login users In fact, one thing I think we should popularize more on Linux is the concept of running multiple unprivileged login users. Personally for the tasks I work on, it often involves building containers or launching local VMs, and isolating that works really well with a full separate "user" identity. An experiment I did was basically useradd ai and running delegated tasks there instead. To log in I added %wheel ALL=NOPASSWD: /usr/bin/machinectl shell ai@ to /etc/sudoers.d/ai-login so that my regular human user could easily get a shell in the ai user's context. I haven't truly "operationalized" this one as juggling separate git repository clones was a bit painful, but I think I could

automate it more. I’m interested in hearing from folks who are doing something similar. Parallel, IaaS-ready agents…with review I’m today often running 2-3 agents in parallel on different tasks (with different levels of success, but that’s its own story). It makes total sense to support delegating some of these agents to work off my local system and into cloud infrastructure. In looking around in this space, there’s quite a lot of stuff. One of them is Ona (formerly Gitpod). I gave it a quick try and I like where they’re going, but more on this below. Github Copilot can also do something similar to this, but what I don’t like about it is that it pushes a model where all of one’s interaction is in the PR. That’s going to be seriously noisy for some repositories, and interaction with LLMs can feel too “personal” sometimes to have permanently recorded. Credentials should be on demand and fine grained for tasks To me a huge flaw with Ona and one shared with other things like Langchain Open-SWE is basically this: Sorry but: no way I’m clicking OK on that button. I need a strong and clearly delineated barrier between tooling/AI agents acting “as me” and my ability to approve and push code or even do basic things like edit existing pull requests. Github’s Copilot gets this more right because its bot runs as a distinct identity. I haven’t dug into what it’s authorized to do. I may play with it more, but I also want to use agents outside of Github and I also am not a fan of deepening dependence on a single proprietary forge either. So I think a key thing agent frontends should help do here is in granting fine-grained ephemeral credentials for dedicated write access as an agent is working on a task. This “credential handling” should be a clearly distinct component. (This goes beyond just git forges of course but also other issue trackers or data sources that may be in context). Conclusion There’s so much out there on this, I can barely keep track while trying to do my real job. I’m sure I’m not alone – but I’m interested in other’s thoughts on this!

- Cassidy James Blaede: I’ve Joined ROOST (2025/10/27 00:00)
  A couple of months ago I shared that I was looking for what was next for me, and I’m thrilled to report that I’ve found it: I’m joining ROOST as OSS Community Manager! What is ROOST? I’ll let our website do most of the talking, but I can add some context based on my conversations with the rest of the incredible ROOSTers over the past few weeks. In a nutshell, ROOST is a relatively new nonprofit focused on building, distributing, and maintaining the open source building blocks for online trust and safety. It was founded by tech industry veterans who saw the need for truly open source tools in the space, and were sick of rebuilding the exact same internal tools across multiple orgs and teams. The way I like to frame it is how you wouldn’t roll your own encryption; why would you roll your own trust and safety tooling? It turns out that currently every platform, service, and community has to reinvent all of the hard work to ensure people are safe and harmful content doesn’t spread. ROOST is teaming up with industry partners to release existing trust and safety tooling as open source and to build the missing pieces together, in the open. The result is that teams will no longer have to start from scratch and take on all of the effort (and risk!) of rolling their own trust and safety tools; instead, they can reach for the open source projects from ROOST to integrate into their own products and systems. And we know open source is the right approach for critical tooling: the tools themselves must be transparent and auditable, while organizations can customize and even help improve this suite of online safety tools to benefit everyone. This Platformer article does a great job of digging into more detail; give it a read. :) Oh, and why the baby chick? ROOST has a habit of naming things after birds—and I’m a baby ROOSTer. :D What is trust and safety? I’ve used the term “trust and safety” a ton in this post; I’m no expert (I’m rapidly learning!), but think about protecting users from harm including unwanted sexual content, misinformation, violent/extremist content, etc. It’s a field that’s much larger in scope and scale than most people probably realize, and is becoming ever more important as it becomes easier to generate massive amounts of text and graphic content using LLMs and related generative “AI” technologies. Add in that those generative technologies are largely trained using opaque data sources that can themselves include harmful content, and you can imagine how we’re at a flash point for trust and safety; robust open online safety tools like those that ROOST is helping to

build and maintain are needed more than ever. What I'll be doing My role is officially "OSS Community Manager," but "community manager" can mean ten different things to ten different people (which is why people in the role often don't survive long at a company…). At ROOST, I feel like the team knows exactly what they need me to do—and importantly, they have a nice onramp and initial roadmap for me to take on! My work will mostly focus on building and supporting an active and sustainable contributor community around our tools, as well as helping improve the discourse and understanding of open source in the trust and safety world. It's an exciting challenge to take on with an amazing team from ROOST as well as partner organizations. My work with GNOME I'll continue to serve on the GNOME Foundation board of directors and contribute to both GNOME and Flathub as much as I can; there may be a bit of a transition time as I get settled into this role, but my open source contributions—both to trust and safety and the desktop Linux world—are super important to me. I'll see you around!

- Aryan Kaushik: Balancing Work and Open Source (2025/10/25 23:00)
  Work pressure + Burnout == Low contributions? Over the past few months, I've been struggling with a tough question. How do I balance my work commitments and personal life while still contributing to open source? On the surface, it looks like a weird question. Like I really enjoy contributing and working with contributors, and when I was in college, I always thought... "Why do people ever step back? It is so fun!". It was the thing that brought a smile to my face and took off any "stress". But now that I have graduated, things have taken a turn. It is now that when work pressure mounts, you use the little time you get to not focus on writing code and instead perform some kind of hobby, learn something new or spend time with family. Or, just endless video scroll and sleep. This has led me to be on my lowest contributions streak and not able to work on all those cool things I imagined, like reworking the Pitivi timeline in Rust, finishing that one MR in GNOME Settings that is stuck for ages, or fixing some issues in GNOME Extensions website, or work on my own extension's feature request, or contributing to the committees I am a part of. It's reached a point where I'm genuinely unsure how to balance things anymore, and hence wanted to give all whom I might not have been able to reply to or have not seen me for a long time an update, that I'm there but just in a dilemma of how to return. I believe I'm not the only one who faces this. After guiding my juniors for a long while on how to contribute and study at the same time and still manage time for other things, I now am at a road where I am in the same situation. So, if anyone has any insights on how they manage their time, or keep up the motivation and juggle between tasks, do let me know (akaushik [at] gnome [dot] org), I'd really appreciate any insights :) One of them would probably be to take fewer things on my plate? Perhaps this is just a new phase of learning? Not about code, but about balance.
- Flathub Blog: Enhanced License Compliance Tools for Flathub (2025/10/24 00:00)
  tl;dr: Flathub has improved tooling to make license compliance easier for developers. Distros should rebuild OS images with updated runtimes from Flathub; app developers should ensure they're using up-to-date runtimes and verify that licenses and copyright notices are properly included. In early August, a concerned community member brought to our attention that copyright notices and license files were being omitted when software was bundled as Flatpaks and distributed via Flathub. This was a genuine oversight across multiple projects, and we're glad we've been able to take the opportunity to correct and improve this for runtimes and apps across the Flatpak ecosystem. Over the past few months, we've been working to enhance our tooling and infrastructure to better support license compliance. With the support of the Flatpak, freedesktop-sdk, GNOME, and KDE teams, we've developed and deployed significant improvements that make it easier than ever for developers to ensure their applications properly include license and copyright notices. What's New In coordination with maintainers of the freedesktop-sdk, GNOME, and KDE runtimes, we've implemented enhanced license handling that automatically includes license and copyright notice files in the runtimes themselves, deduplicated to be as space-efficient as possible. This improvement has been applied to all supported freedesktop-sdk, GNOME, and

KDE runtimes, plus backported to freedesktop-sdk 22.08 and newer, GNOME 45 and newer, KDE 5.15-22.08 and newer, and KDE 6.6 and newer. These updated runtimes cover over 90% of apps on Flathub and have already rolled out to users as regular Flatpak updates. We've also worked with the Flatpak developers to add new functionality to flatpak-builder 1.4.5 that automatically recognizes and includes common license files. This enhancement, now deployed to the Flathub build service, helps ensure apps' own licenses as well as the licenses of any bundled libraries are retained and shipped to users along with the app itself. These improvements represent an important milestone in the maturity of the Flatpak ecosystem, making license compliance easier and more automatic for the entire community. Recommended Actions App Developers We encourage you to rebuild your apps with flatpak-builder 1.4.5 or newer to take advantage of the new automatic license detection. You can verify that license and copyright notices are properly included in your Flatpak's /app/share/licenses, both for your app and any included dependencies. In most cases, simply rebuilding your app will automatically include the necessary licenses, but you can also fine-tune which license files are included using the license-files key in your app's Flatpak manifest if needed. For apps with binary sources (e.g. debs or rpms), we encourage app maintainers to explicitly include relevant license files in the Flatpak itself for consistency and auditability. End-of-life runtime transition: To focus our resources on maintaining high-quality, up-to-date runtimes, we'll be completing the removal of several end-of-life runtimes in January 2026. Apps using runtimes older than freedesktop-sdk 22.08, GNOME 45, KDE 5.15-22.08 or KDE 6.6 will be marked as EOL shortly. Once these older runtimes are removed, the apps will need to be updated to use a supported runtime to remain available on Flathub. While this won't affect existing app installations, after this date, new users will be unable to install these apps from Flathub until they're rebuilt against a current runtime. Flatpak manifests of any affected apps will remain on the Flathub GitHub organization to enable developers to update them at any time. If your app currently targets an end-of-life runtime that did receive the backported license improvements, we still strongly encourage you to upgrade to a newer, supported runtime to benefit from ongoing security updates and platform improvements. Distributors If you redistribute binaries from Flathub, such as pre-installed runtimes or apps, you should rebuild your distributed images (ISOs, containers, etc.) with the updated runtimes and apps from Flathub. You can verify that appropriate licenses are included with the Flatpaks in the runtime filesystem at /usr/share/licenses inside each runtime. Get in Touch App developers, distributors, and community members are encouraged to connect with the team and other members of the community in our Discourse forum and Matrix chat room. If you are an app developer or distributor and have any questions or concerns, you may also reach out to us at admins@flathub.org. Thank You! We are grateful to Jef Spaleta from Fedora for his care and confidentiality in bringing this to our attention and working with us collaboratively throughout the process. Special thanks to Boudhayan Bhattcharya (bbhtt) for his tireless work across Flathub, Flatpak and freedesktop-sdk, on this as well as many other important areas. And thank you to Abderrahim Kitouni (akitouni), Adrian Vovk (AdrianVovk), Aleix Pol Gonzalez (apol), Bart Piotrowski (barthalion), Ben Cooksley (bcooksley), Javier Jardón (jjardon), Jordan Petridis (alatiera), Matthias Clasen (matthiasc), Rob McQueen (ramcq), Sebastian Wick (swick), Timothée Ravier (travier), and any others behind the scenes for their hard work and timely collaboration across multiple projects to deliver these improvements. Our Linux app ecosystem is truly strongest when individuals from across companies and projects come together to collaborate and work towards shared goals. We look forward to continuing to work together to ensure app developers can easily ship their apps to users across all Linux distributions and desktop environments. ♥

- Matthias Clasen: SVG in GTK (2025/10/23 10:34)
  GTK has been using SVG for symbolic icons since essentially forever. It hasn’t been a perfect relationship, though. Pre-History For the longest time (all through the GTK 3 era, and until recently), we’ve used librsvg indirectly, through gdk-pixbuf, to obtain rendered icons, and then we used

some pixel tricks to recolor the resulting image according to the theme. This works, but it gives up on the defining feature of SVG: its scalability. Once you've rasterized your icon at a given size, all you're left with is pixels. In the GTK 3 era, this wasn't a problem, but in GTK 4, we have a scene graph and fractional scaling, so we could do *much* better if we don't rasterize early. Unfortunately, librsvg's API isn't set up to let us easily translate SVG into our own render nodes. And its rust nature makes for an inconvenient dependency, so we held off on depending on it for a long time. History Early this year, I grew tired of this situation, and decided to improve our story for icons, and symbolic ones in particular. So I set out to see how hard it would be to parse the very limited subset of SVG used in symbolic icons myself. It turned out to be relatively easy. I quickly managed to parse 99% of the Adwaita symbolic icons, so I decided to merge this work for GTK 4.20. There were some detours and complications along the way. Since my simple parser couldn't handle 100% of Adwaita (let alone all of the SVGs out there), a fallback to a proper SVG parser was needed. So we added a librsvg dependency after all. Since our new Android backend has an even more difficult time with rust than our other backends, we needed to arrange for a non-rust librsvg branch to be used when necessary. One thing that this hand-rolled SVG parser improved upon is that it allows stroking, in addition to filling. I documented the format for symbolic icons here. Starting over A bit later, I was inspired by Apple's SF Symbols work to look into how hard it would be to extend my SVG parser with a few custom attributes to enable dynamic strokes. It turned out to be easy again. With a handful of attributes, I could create plausible-looking animations and transitions. And it was fun to play with. When I showed this work to Jakub and Lapo at GUADEC, they were intrigued, so I decided to keep pushing this forward, and it landed in early GTK 4.21, as GtkPathPaintable. https://blogs.gnome.org/gtk/files/2025/10/path-anim5.webm To make experimenting with this easier, I made a quick editor. It was invaluable to have Jakub as an early adopter play with the editor while I was improving the implementation. Some very good ideas came out of this rapid feedback cycle, for example dynamic stroke width. You can get some impression of the new stroke-based icons Jakub has been working on here. Recent happenings As summer was turning to fall, I felt that I should attempt to support SVG more completely, including grouping and animations. GTK's rendering infrastructure has most of the pieces that are required for SVG after all: transforms, filters, clips, paths, gradients are all supported. This was *not* easy. But eventually, things started to fall into place. And this week, I've replaced GtkPathPaintable with GtkSvg, which is a GdkPaintable that supports SVG. At least, the subset of SVG that is most relevant for icons. And that includes animations. https://blogs.gnome.org/gtk/files/2025/10/Screencast-From-2025-10-22-18-13-02.webm This is still a subset of full SVG, but converting a few random lottie files to SVG animations gave me a decent success rate for getting things to display mostly ok. The details are spelled out here. Summary GTK 4.22 will natively support SVG, including SVG animations. If you'd like to help improve this further, here are some some suggestions. If you would like to support the GNOME foundation, who's infrastructure and hosting GTK relies on, please donate.

- Jonathan Blandford: Crosswords 0.3.16: 2025 Internship Results (2025/10/23 06:00)
  Time for another GNOME Crosswords release! This one highlights the features our interns did this past summer. We had three fabulous interns — two through GSoC and one through Outreachy. While this release really only has three big features — one from each — they were all fantastic. Thanks goes to to my fellow GSoC mentors Federico and Tanmay. In addition, Tilda and the folks at Outreachy were extremely helpful. Mentorship is a lot of work, but it's also super-rewarding. If you're interested in participating as a mentor in the future and have any questions about the process, let me know. I'll be happy to speak with you about them. Dictionary pipeline improvements First, our Outreachy intern Nancy spent the summer improving the build pipeline to generate the internal dictionaries we use. These dictionaries are used to provide autofill capabilities and add definitions to the Editor, as well as providing near-instant completions for both the Editor and Player. The old pipeline was

buggy and hard to maintain. Once we had a cleaned it up, Nancy was able to use it to effortlessly produce a dictionary in her native tongue: Swahili. A Grid in swahili We have no distribution story yet, but it's exciting to have it so much easier to create dictionaries in other languages. It opens the door to the Editor being more universally useful (and fulfills a core GNOME tenet). You can read about it more details in Nancy's final report. Word List Victor did a ton of research for Crosswords, almost acting like a Product Manager. He installed every crossword editor he could find and did a competitive analysis, noting possible areas for improvement. One of the areas he flagged was the word list in our editor. This list suggests words that could be used in a given spot in the grid. We started with a simplistic implementation that listed every possible word in our dictionary that could fit. This approach— while fast — provided a lot of dead words that would make the grid unsolvable. So he set about trying to narrow down that list. New Word List showing possible options It turns out that there's a lot of tradeoffs to be made here (Victor's post). It's possible to find a really good set of words, at the cost of a lot of computational power. A much simpler list is quick but has dead words. In the end, we found a happy medium that let us get results fast and had a stable list across a clue. He'll be blogging about this shortly. Victor also cleaned up our development docs, and researched satsolve algorithms for the grid. He's working on a lovely doc on the AC-3 algorithm, and we can use it to add additional functionality to the editor in the future. Printing Toluwaleke implemented printing support for GNOME Crosswords. This was a tour de force, and a phenomenal addition to the Crosswords codebase. When I proposed it for a GSoC project, I had no idea how much work this project could involve. We already had code to produce an svg of the grid — I thought that we could just quickly add support for the clues and call it a day. Instead, we ended up going on a wild ride resulting in a significantly stronger feature and code base than we had going in. His blog has more detail and it's really quite cool (go read it!). But from my perspective, we ended up with a flexible and fast rendering system that can be used in a lot more places. Take a look: https://blogs.gnome.org/jrb/files/2025/10/output_video.webm The resulting PDFs are really high quality — they seem to look better than some of the newspaper puzzles I've seen. We'll keep tweaking them as there are still a lot of improvements we'd like to add, such as taking the High Contrast / Large Text A11Y options into account. But it's a tremendous basis for future work. Increased Polish There were a few other small things that happened I hooked Crosswords up to Damned Lies. This led to an increase in our translation quality and count This included a Polish translation, which came with a new downloader! I ported all the dialogs to AdwDialog, and moved on from (most) of the deprecated Gtk4 widgets A lot of code cleanups and small fixes Now that these big changes have landed, it's time to go back to working on the rest of the changes proposed for GNOME Circle. Until next time, happy puzzling!

- Toluwaleke Ogundipe: GSoC Final Report: Printing in GNOME Crosswords (2025/10/22 22:50)
  A few months ago, I introduced my GSoC project: Adding Printing Support to GNOME Crosswords. Since June, I've been working hard on it, and I'm happy to share that printing puzzles is finally possible! The Result GNOME Crosswords now includes a Print option in its menu, which opens the system's print dialog. After adjusting printer settings and page setup, the user is shown a preview dialog with a few crossword-specific options, such as ink-saving mode and whether (and how) to include the solution. The options are intentionally minimal, keeping the focus on a clean and straightforward printing experience. Below is a short clip showing the feature in action: The resulting file: output.pdf Crosswords now also ships with a standalone command-line tool, ipuz2pdf, which converts any IPUZ puzzle file into a print-ready PDF. It offers a similarly minimal set of layout and crossword-specific options. The Process Studied and profiled the existing code and came up with an overall approach for the project. Built a new grid rendering framework, resulting in a 10× speedup in rendering. Dealt with a ton of details around text placement and rendering, colouring, shapes, and more. Designed and implemented a print layout engine with a templating system, adjusted to work with different puzzle kinds, grid sizes, and paper sizes. Integrated the layout engine with the print dialog and added a live print preview. Bonus:

Created ipuz2pdf, a standalone command-line utility (originally for testing) that converts an IPUZ file into a printable PDF. The Challenges Working on a feature of this scale came with plenty of challenges. Getting familiar with a large codebase took patience, and understanding how everything fit together often meant careful study and experimentation. Balancing ideas with the project timeline and navigating code reviews pushed me to grow both technically and collaboratively. On the technical side, rendering and layout had their own hurdles. Handling text metrics, scaling, and coordinate transformations required a mix of technical knowledge, critical thinking, and experimentation. Even small visual glitches could lead to hours of debugging. One notably difficult part was implementing the box layout system that powers the dynamic print layout engine. The Lessons This project taught me a lot about patience, focus, and iteration. I learned to approach large problems by breaking them into small, testable pieces, and to value clarity and simplicity in both code and design. Code reviews taught me to communicate ideas better, accept feedback gracefully, and appreciate different perspectives on problem-solving. On the technical side, working with rendering and layout systems deepened my understanding of graphics programming. I also learned how small design choices can ripple through an entire codebase, and how careful abstraction and modularity can make complex systems easier to evolve. Above all, I learned the value of collaboration, and that progress in open source often comes from many small, consistent improvements rather than big leaps. The Conclusion In the end, I achieved all the goals set out for the project, and even more. It was a long and taxing journey, but absolutely worth it. The Gratitude I’m deeply grateful to my mentors, Jonathan Blandford and Federico Mena Quintero, for their guidance, patience, and support throughout this project. I’ve learned so much from working with them. I’m also grateful to the GNOME community and Google Summer of Code for making this opportunity possible and for creating such a welcoming environment for new contributors. What Comes After No project is ever truly finished, and this one is no exception. There’s still plenty to be done, and some already have tracking issues. I plan to keep improving the printing system and related features in GNOME Crosswords. I also hope to stay involved in the GNOME ecosystem and open-source development in general. I’m especially interested in projects that combine design, performance, and system-level programming. More importantly, I’m a recent CS graduate looking for a full-time role in the field of interest stated earlier. If you have or know of any opportunities, please reach out to me. Finally, I plan to write a couple of follow-up posts diving into interesting parts of the process in more detail. Stay tuned! Thank you!

- Dorothy Kabarozi: Laravel Mix “Unable to Locate Mix File” Error: Causes and Fixes (2025/10/21 16:40)
  Laravel Mix “Unable to Locate Mix File” Error: Causes and Fixes If you’re working with Laravel and using Laravel Mix to manage your CSS and JavaScript assets, you may have come across an error like this: Spatie\LaravelIgnition\Exceptions\ViewException Message: Unable to locate Mix file: /assets/vendor/css/rtl/core.css Or in some cases: Illuminate\Foundation\MixFileNotFoundException Unable to locate Mix file: /assets/vendor/fonts/boxicons.css This error can be frustrating, especially when your project works perfectly on one machine but fails on another. Let’s break down what’s happening and how to solve it. What Causes This Error? Laravel Mix is a wrapper around Webpack, designed to compile your resources/ assets (CSS, JS, images) into the public/ directory. The mix() helper in Blade templates references these compiled assets using a special file: mix-manifest.json. This error occurs when Laravel cannot find the compiled asset. Common reasons include: Assets are not compiled yetIf you’ve just cloned a project, the public/assets folder might be empty. Laravel is looking for files that do not exist yet. mix-manifest.json is missing or outdatedThis file maps original asset paths to compiled paths. If it’s missing, Laravel Mix won’t know where to find your assets. Incorrect paths in Blade templatesIf your code is like: <link rel="stylesheet" href="{{ asset(mix('assets/vendor/css/rtl/core.css')) }}" /> but the RTL folder or the file doesn’t exist, Mix will throw an exception. Wrong configurationSome themes use variables like $configData['rtlSupport'] to toggle right-to-left CSS. If it’s set incorrectly, Laravel will try to load files that don’t exist. How to Fix It Here’s a step-by-step solution: 1. Install

NPM dependencies Make sure you have Node.js installed, then run: npm install 2. Compile your assets Development mode (fast, unminified): npm run dev Production mode (optimized, minified): npm run build This will generate your CSS and JS files in the public folder and update mix-manifest.json. 3. Check mix-manifest.json Ensure the manifest contains the file Laravel is looking for: "/assets/vendor/css/rtl/core.css": "/assets/vendor/css/rtl/core.css" 4. Adjust Blade template paths If you don’t use RTL, you can set: $configData['rtlSupport'] = ''; so the code doesn’t try to load /rtl/core.css unnecessarily. 5. Clear caches Laravel may cache old views and configs. Clear them: php artisan view:clear php artisan config:clear php artisan cache:clear Pro Tips Always check if the file exists in public/assets/... after compiling. If you move your project to another machine or server, you must run npm install and npm run dev again. For production, make sure your server has Node.js and NPM installed, otherwise Laravel Mix cannot build the assets. Conclusion The “Unable to locate Mix file” error is not a bug in Laravel, but a result of missing compiled assets or misconfigured paths. Once you: Install dependencies. Compile assets, Correct Blade paths, and Clear caches; your Laravel project should load CSS and JS files without issues.

- Daniel García Moreno: GNOME Tour in openSUSE and welcome app (2025/10/21 12:00)
  As a follow up of the Hackweek 24 project, I've continued working on the gnome-tour fork for openSUSE with custom pages to replace the welcome application for openSUSE distributions. GNOME Tour modifications All the modifications are on top of upstream gnome-tour and stored in the openSUSE/gnome-tour repo Custom initial page A new donations page. In openSUSE we remove the popup from GNOME shell for donations, so it's fair to add it in this place. Last page with custom openSUSE links, this one is the used for opensuse-welcome app. opensuse-welcome package The original opensuse-welcome is a qt application, and this one is used for all desktop environments, but it's more or less unmaintained and looking for a replacement, we can use the gnome-tour fork as the default welcome app for all desktop without a custom app. To do a minimal desktop agnostic opensuse-welcome application, I've modified the gnome-tour to also generate a second binary but just with the last page. The new opensuse-welcome rpm package is built as a subpackage of gnome-tour. This new application is minimal and it doesn't have lots of requirements, but as it's a gtk4 application, it requires gtk and libadwaita, and also depends on gnome-tour-data to get the resoures of the app. To improve this welcome app we need to review the translations, because I added three new pages to the gnome-tour and that specific pages are not translated, so I should regenerate the .po files for all languages and upload to openSUSE Weblate for translations.
- Matthew Garrett: Where are we on X Chat security? (2025/10/20 23:36)
  AWS had an outage today and Signal was unavailable for some users for a while. This has confused some people, including Elon Musk, who are concerned that having a dependency on AWS means that Signal could somehow be compromised by anyone with sufficient influence over AWS (it can't). Which means we're back to the richest man in the world recommending his own "X Chat", saying The messages are fully encrypted with no advertising hooks or strange “AWS dependencies” such that I can’t read your messages even if someone put a gun to my head.Elon is either uninformed about his own product, lying, or both.As I wrote back in June, X Chat genuinely end-to-end encrypted, but ownership of the keys is complicated. The encryption key is stored using the Juicebox protocol, sharded between multiple backends. Two of these are asserted to be HSM backed - a discussion of the commissioning ceremony was recently posted here. I have not watched the almost 7 hours of video to verify that this was performed correctly, and I also haven't been able to verify that the public keys included in the post were the keys generated during the ceremony, although that may be down to me just not finding the appropriate point in the video (sorry, Twitter's video hosting doesn't appear to have any skip feature and would frequently just sit spinning if I tried to seek to far and I should probably just download them and figure it out but I'm not doing that now). With enough effort it would probably also have been possible to fake the entire thing - I have no reason to believe that

this has happened, but it's not externally verifiable.But let's assume these published public keys are legitimately the ones used in the HSM Juicebox realms[1] and that everything was done correctly. Does that prevent Elon from obtaining your key and decrypting your messages? No.On startup, the X Chat client makes an API call called GetPublicKeysResult, and the public keys of the realms are returned. Right now when I make that call I get the public keys listed above, so there's at least some indication that I'm going to be communicating with actual HSMs. But what if that API call returned different keys? Could Elon stick a proxy in front of the HSMs and grab a cleartext portion of the key shards? Yes, he absolutely could, and then he'd be able to decrypt your messages.(I will accept that there is a plausible argument that Elon is telling the truth in that even if you held a gun to his head he's not smart enough to be able to do this himself, but that'd be true even if there were no security whatsoever, so it still says nothing about the security of his product)The solution to this is remote attestation - a process where the device you're speaking to proves its identity to you. In theory the endpoint could attest that it's an HSM running this specific code, and we could look at the Juicebox repo and verify that it's that code and hasn't been tampered with, and then we'd know that our communication channel was secure. Elon hasn't done that, despite it being table stakes for this sort of thing (Signal uses remote attestation to verify the enclave code used for private contact discovery, for instance, which ensures that the client will refuse to hand over any data until it's verified the identity and state of the enclave). There's no excuse whatsoever to build a new end-to-end encrypted messenger which relies on a network service for security without providing a trustworthy mechanism to verify you're speaking to the real service.We know how to do this properly. We have done for years. Launching without it is unforgivable.[1] There are three Juicebox realms overall, one of which doesn't appear to use HSMs, but you need at least two in order to obtain the key so at least part of the key will always be held in HSMs comments

- Dorothy Kabarozi: Deploying a Simple HTML Project on Linode Using Nginx (2025/10/18 16:14)
  Deploying a Simple HTML Project on Linode Using Nginx: My Journey and Lessons Learned Deploying web projects can seem intimidating at first, especially when working with a remote server like Linode. Recently, I decided to deploy a simple HTML project (index.html) on a Linode server using Nginx. Here’s a detailed account of the steps I took, the challenges I faced, and the solutions I applied. Step 1: Accessing the Linode Server The first step was to connect to my Linode server via SSH: ssh root@<your-linode-ip> Initially, I encountered a timeout issue, which reminded me to check network settings and ensure SSH access was enabled for my Linode instance. Once connected, I had access to the server terminal and could manage files and services. Step 2: Preparing the Project My project was simple—it only contained an index.html file. I uploaded it to the server under: /var/www/hng13-stage0-devops I verified the project folder structure with: ls -l /var/www/hng13-stage0-devops Since there was no public folder or PHP files, I knew I needed to adjust the Nginx configuration to serve directly from this folder. Step 3: Setting Up Nginx I opened the Nginx configuration for my site: sudo nano /etc/nginx/sites-available/hng13 Initially, I mistakenly pointed root to a non-existent folder (public), which caused a 404 Not Found error. The correct configuration looked like this: server { listen 80; server_name <your_linode-ip>; root /var/www/hng13-stage0-devops; # points to folder containing index.html index index.html index.htm; location / { try_files $uri $uri/ =404; } } Step 4: Enabling the Site and Testing After creating the configuration file, I enabled the site: sudo ln -s /etc/nginx/sites-available/hng13 /etc/nginx/sites-enabled/ I also removed the default site to avoid conflicts: sudo rm /etc/nginx/sites-enabled/default Then I tested the configuration: sudo nginx -t If the syntax was OK, I reloaded Nginx: sudo systemctl reload nginx Step 5: Checking Permissions Nginx must have access to the project files. I ensured the correct permissions: sudo chown -R www-data:www-data /var/www/hng13-stage0-devops sudo chmod -R 755 /var/www/hng13-stage0-devops Step 6: Viewing the Site Finally, I opened my browser and navigated to http://<your-linode-ip> And there it was—my index.html page served perfectly via Nginx. Challenges and Lessons Learned Nginx server_name Error Error: "server_name" directive is

not allowed here Lesson: Always place server_name inside a server { ... } block. 404 Not Found Cause: Nginx was pointing to a public folder that didn’t exist. Solution: Update root to the folder containing index.html. Permissions Issues Nginx could not read files initially. Solution: Ensure ownership by www-data and proper read/execute permissions. SSH Timeout / Connection Issues Double-check firewall rules and Linode network settings. Key Takeaways For static HTML projects, Nginx is simple and effective. Always check the root folder matches your project structure. Testing the Nginx config (nginx -t) before reload saves headaches. Proper permissions are crucial for serving files correctly. Deploying my project was a learning experience. Even small mistakes like pointing to the wrong folder or placing directives in the wrong context can break the site—but step-by-step debugging and understanding the errors helped me fix everything quickly.This has kick started my devOps journey and I truly loved the challenge

- Gedit Technology blog: Mid-October News (2025/10/15 10:00)
  Misc news about the gedit text editor, mid-October edition! (Some sections are a bit technical). Rework of the file loading and saving (continued) The refactoring continues in the libgedit-gtksourceview module, this time to tackle a big class that takes too much responsibilities. A utility is in development which will permit to delegate a part of the work. The utility is about character encoding conversion, with support of invalid bytes. It takes as input a single GBytes (the file content), and transforms it into a list of chunks. A chunk contains either valid (successfully converted) bytes, or invalid bytes. The output format - the "list of chunks" - is subject to change to improve memory consumption and performances. Note that invalid bytes are allowed, to be able to open really any kind of files with gedit. I must also note that this is quite sensitive work, at the heart of document loading for gedit. Normally all these refactorings and improvements will be worth it! Progress in other modules There has been some progress on other modules: gedit: version 48.1.1 has been released with a few minor updates. The Flatpak on Flathub: update to gedit 48.1.1 and the GNOME 49 runtime. gspell: version 1.14.1 has been released, mainly to pick up the updated translations. GitHub Sponsors In addition to Liberapay, you can now support the work that I do on GitHub Sponsors. See the gedit donations page. Thank you ❤
- Victor Ma: This is a test post (2025/10/15 00:00)
  Over the past few weeks, I’ve been working on improving some test code that I had written. Refactoring time! My first order of business was to refactor the test code. There was a lot of boilerplate, which made it difficult to add new tests, and also created visual clutter. For example, have a look at this test case: static void test_egg_ipuz (void) { g_autoptr (WordList) word_list = NULL; IpuzGrid *grid; g_autofree IpuzClue *clue = NULL; g_autoptr (WordArray) clue_matches = NULL; word_list = get_broda_word_list (); grid = create_grid (EGG_IPUZ_FILE_PATH); clue = get_clue (grid, IPUZ_CLUE_DIRECTION_ACROSS, 2); clue_matches = word_list_find_clue_matches (word_list, clue, grid); g_assert_cmpint (word_array_len (clue_matches), ==, 3); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 0)), ==, "EGGS"); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 1)), ==, "EGGO"); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 2)), ==, "EGGY"); } That’s an awful lot of code just to say: Use the EGG_IPUZ_FILE_PATH file. Run the word_list_find_clue_matches() function on the 2-Across clue. Assert that the results are ["EGGS", "EGGO", "EGGY"]. And this was repeated in every test case, and needed to be repeated in every new test case I added. So, I knew that I had to refactor my code. Fixtures and functions My first step was to extract all of this setup code: g_autoptr (WordList) word_list = NULL; IpuzGrid *grid; g_autofree IpuzClue *clue = NULL; g_autoptr (WordArray) clue_matches = NULL; word_list = get_broda_word_list (); grid = create_grid (EGG_IPUZ_FILE_PATH); clue = get_clue (grid, IPUZ_CLUE_DIRECTION_ACROSS, 2); clue_matches = word_list_find_clue_matches (word_list, clue, grid); To do this, I used a fixture: typedef struct { WordList *word_list; IpuzGrid *grid; } Fixture; static void fixture_set_up (Fixture *fixture,

gconstpointer user_data) { const gchar *ipuz_file_path = (const gchar *) user_data; fixture->word_list = get_broda_word_list (); fixture->grid = create_grid (ipuz_file_path); } static void fixture_tear_down (Fixture *fixture, gconstpointer user_data) { g_object_unref (fixture->word_list); } My next step was to extract all of this assertion code: g_assert_cmpint (word_array_len (clue_matches), ==, 3); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 0)), ==, "EGGS"); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 1)), ==, "EGGO"); g_assert_cmpstr (word_list_get_indexed_word (word_list, word_array_index (clue_matches, 2)), ==, "EGGY"); To do this, I created a new function that runs word_list_find_clue_matches() and asserts that the result equals an expected_words parameter. static void test_clue_matches (WordList *word_list, IpuzGrid *grid, IpuzClueDirection clue_direction, guint clue_index, const gchar *expected_words[]) { const IpuzClue *clue = NULL; g_autoptr (WordArray) clue_matches = NULL; g_autoptr (WordArray) expected_word_array = NULL; clue = get_clue (grid, clue_direction, clue_index); clue_matches = word_list_find_clue_matches (word_list, clue, grid); expected_word_array = str_array_to_word_array (expected_words, word_list); g_assert_true (word_array_equals (clue_matches, expected_word_array)); } After all that, here's what my test case looked like: static void test_egg_ipuz (Fixture *fixture, gconstpointer user_data) { test_clue_matches (fixture->word_list, fixture->grid, IPUZ_CLUE_DIRECTION_ACROSS, 2, (const gchar*[]){"EGGS", "EGGO", "EGGY", NULL}); } Much better! Macro functions But as great as that was, I knew that I could take it even further, with macro functions. I created a macro function to simplify test case definitions: #define ASSERT_CLUE_MATCHES(DIRECTION, INDEX, ...) \ test_clue_matches (fixture->word_list, \ fixture->grid, \ DIRECTION, \ INDEX, \ (const gchar*[]){__VA_ARGS__, NULL}) Now, test_egg_ipuz() looked like this: static void test_egg_ipuz (Fixture *fixture, gconstpointer user_data) { ASSERT_CLUE_MATCHES (IPUZ_CLUE_DIRECTION_ACROSS, 2, "EGGS", "EGGO", "EGGY"); } I also made a macro function for the test case declarations: #define ADD_IPUZ_TEST(test_name, file_name) \ g_test_add ("/clue_matches/" #test_name, \ Fixture, \ "tests/clue-matches/" #file_name, \ fixture_set_up, \ test_name, \ fixture_tear_down) Which turned this: g_test_add ("/clue_matches/test_egg_ipuz", Fixture, EGG_IPUZ, fixture_set_up, test_egg_ipuz, fixture_tear_down); Into this: ADD_IPUZ_TEST (test_egg_ipuz, egg.ipuz); An unfortunate bug So, picture this: You've just finished refactoring your test code. You add some finishing touches, do a final test run, look over the diff one last time…and everything seems good. So, you open up an MR and start working on other things. But then, the unthinkable happens—the CI pipeline fails! And apparently, it's due to a test failure? But you ran your tests locally, and everything worked just fine. (You run them again just to be sure, and yup, they still pass.) And what's more, it's only the Flatpak CI tests that failed. The native CI tests succeeded. So…what, then? What could be the cause of this? I mean, how do you even begin debugging a test failure that only happens in a particular CI job and nowhere else? Well, let's just try running the CI pipeline again and see what happens. Maybe the problem will go away. Hopefully, the problem goes away. … Nope. Still fails. … Rats. Well, I'll spare you the gory details that it took for me to finally figure this one out. But the cause of the bug was me accidentally freeing an object that I should never have freed. This meant that the corresponding memory segment could be—but, importantly, did not necessarily have to be—filled with garbage data. And this is why only the Flatpak job's test run failed…well, at first, anyway. By changing around some of the test cases, I was able to get the native CI tests and local tests to fail. And this is what eventually clued me into the true nature of this bug. So, after spending the better part of two weeks, here is the fix I ended up with: @@ -94,7 +94,7 @@ test_clue_matches (WordList *word_list, guint clue_index, const gchar *expected_words[]) { - g_autofree IpuzClue *clue = NULL; + const IpuzClue *clue = NULL; g_autoptr (WordArray) clue_matches = NULL; g_autoptr (WordArray) expected_word_array = NULL;

- Jordan Petridis: Nightly Flatpak CI gets a cache (2025/10/14 18:00)
  Recently I got around tackling a long standing issue for good. There were multiple attempts in the past 6 years to cache flatpak-builder artifacts

with Gitlab but none had worked so far. On the technical side of things, flatpak-builder relies heavily on extended attributes (xattrs) on files to do cache validation. Using gitlab's built-in cache or artifacts mechanisms results in a plain zip archive which strips all the attributes from the files, causing the cache to always be invalid once restored. Additionally the hardlinks/symlinks in the cache break. One workaround for this is to always tar the directories and then manually extract them after they are restored. On the infrastructure of things we stumble once again into Gitlab. When a cache or artifact is created, it's uploaded into the Gitlab's instance storage so it can later be reused/redownloaded into any runner. While this is great, it also quickly ramps up the network egress bill we have to pay along with storage. And since its a public gitlab instance that anyone can make request against repositories, it gets out of hand fast. Couple weeks ago Bart pointed me out to Flathub's workaround for this same problem. It comes down to making it someone else problem, and ideally one someone who is willing to fund FOSS infrastructure. We can use ORAS to wrap files and directories into an OCI wrapper and publish it to public registries. And it worked. Quite handy! OCI images are the new tarballs. Now when a pipeline run against your default branch (and assuming it's protected) it will create a cache artifact and upload to the currently configured OCI registry. Afterwards, any build, including Merge Request pipelines, will download the image, extract the artifacts and check how much of it is still valid. From some quick tests and numbers, GNOME Builder went from a ~16 minute build to 6 minutes for our x86_64 runners. While on the AArch64 runner the impact was even bigger, going from 50 minutes to 16 minutes. Not bad. The more modules you are building in your manifest, the more noticeable it is. Unlike Buildstream, there is no Content Addressable Server and flatpak-builder itself isn't aware of the artifacts we publish or can associate them with the cache keys. The OCI/ORAS cache artifacts are manual and a bit hacky of a solution but works well in practice and until we have better tooling. To optimize a bit better for less cache-misses consider building modules from pinned commits/tags/tarballs and building modules from moving branches as late as possible. If you are curious in the details, take a look at the related Merge Request in the templates repository and the follow up commits. Free Palestine

- Bilal Elmoussaoui: Testing a Rust library - Code Coverage (2025/10/13 00:00)
  It has been a couple of years since I started working on a Rust library called oo7 as a Secret Service client implementation. The library ended up also having support for per-sandboxed app keyring using the Secret portal with a seamless API for end-users that makes usage from the application side straightforward. The project, with time, grew support for various components: oo7-cli: A secret-tool replacement but much better, as it allows not only interacting with the Secret service on the DBus session bus but also with any keyring. oo7-cli --app-id com.belmoussaoui.Authenticator list, for example, allows you to read the sandboxed app with app-id com.belmoussaoui.Authenticator's keyring and list its contents, something that is not possible with secret-tool. oo7-portal: A server-side implementation of the Secret portal mentioned above. Straightforward, thanks to my other library ASHPD. cargo-credential-oo7: A cargo credential provider built using oo7 instead of libsecret. oo7-daemon: A server-side implementation of the Secret service. The last component was kickstarted by Dhanuka Warusadura, as we already had the foundation for that in the client library, especially the file backend reimplementation of gnome-keyring. The project is slowly progressing, but it is almost there! The problem with replacing such a very sensitive component like gnome-keyring-daemon is that you have to make sure the very sensitive user data is not corrupted, lost, or inaccessible. For that, we need to ensure that both the file backend implementation in the oo7 library and the daemon implementation itself are well tested. That is why I spent my weekend, as well as a whole day off, working on improving the test suite of the wannabe core component of the Linux desktop. Coverage Report One metric that can give the developer some insight into which lines of code or functions of the codebase are executed when running the test suite is code coverage. In order to get the coverage of a Rust project, you can use a project like Tarpaulin, which integrates with the Cargo build system. For a simple project, a command

like this, after installing Tarpaulin, can give you an HTML report: cargo tarpaulin \ --package oo7 \ --lib \ --no-default-features \ --features "tracing,tokio,native_crypto" \ --ignore-panics \ --out Html \ --output-dir coverage Except in our use case, it is slightly more complicated. The client library supports switching between Rust native cryptographic primitives crates or using OpenSSL. We must ensure that both are tested. For that, we can export our report in LCOV for native crypto and do the same for OpenSSL, then combine the results using a tool like grcov. mkdir -p coverage-raw cargo tarpaulin \ --package oo7 \ --lib \ --no-default-features \ --features "tracing,tokio,native_crypto" \ --ignore-panics \ --out Lcov \ --output-dir coverage-raw mv coverage-raw/lcov.info coverage-raw/native-tokio.info cargo tarpaulin \ --package oo7 \ --lib \ --no-default-features \ --features "tracing,tokio,openssl_crypto" \ --ignore-panics \ --out Lcov \ --output-dir coverage-raw mv coverage-raw/lcov.info coverage-raw/openssl-tokio.info and then combine the results with cat coverage-raw/*.info > coverage-raw/combined.info grcov coverage-raw/combined.info \ --binary-path target/debug/ \ --source-dir . \ --output-type html \ --output-path coverage \ --branch \ --ignore-not-existing \ --ignore "**/portal/*" \ --ignore "**/cli/*" \ --ignore "**/tests/*" \ --ignore "**/examples/*" \ --ignore "**/target/*" To make things easier, I added a bash script to the project repository that generates coverage for both the client library and the server implementation, as both are very sensitive and require intensive testing. With that script in place, I also used it on CI to generate and upload the coverage reports at https://bilelmoussaoui.github.io/oo7/coverage/. The results were pretty bad when I started. Testing For the client side, most of the tests are straightforward to write; you just need to have a secret service implementation running on the DBus session bus. Things get quite complicated when the methods you have to test require a Prompt, a mechanism used in the spec to define a way for the user to be prompted for a password to unlock the keyring, create a new collection, and so on. The prompter is usually provided by a system component. For now, we just skipped those tests. For the server side, it was mostly about setting up a peer-to-peer connection between the server and the client: let guid = zbus::Guid::generate(); let (p0, p1) = tokio::net::UnixStream::pair().unwrap(); let (client_conn, server_conn) = tokio::try_join!(// Client zbus::connection::Builder::unix_stream(p0).p2p().build(), // Server zbus::connection::Builder::unix_stream(p1) .server(guid) .unwrap() .p2p() .build(),) .unwrap(); Thanks to the design of the client library, we keep the low-level APIs under oo7::dbus::api, which allowed me to straightforwardly write a bunch of server-side tests already. There are still a lot of tests that need to be written and a few missing bits to ensure oo7-daemon is in an acceptable shape to be proposed as an alternative to gnome-keyring. Don't overdo it The coverage report is not meant to be targeted at 100%. It’s not a video game. You should focus only on the critical parts of your code that must be tested. Testing a Debug impl or a From trait (if it is straightforward) is not really useful, other than giving you a small dose of dopamine from "achieving" something. Till then, may your coverage never reach 100%.

- Hubert Figuière: Dev Log September 2025 (2025/10/11 00:00)
  Not as much as I wanted to do was done in September. libopenraw Extracting more of the calibration values for colour correction on DNG. Currently work on fixing the purple colour cast. Added Nikon ZR and EOS C50. ExifTool Submitted some metadata updates to ExifTool. Because it nice to have, and also because libopenraw uses some of these autogenerated: I have a Perl script to generate Rust code from it (it used to do C++). Niepce Finally merged the develop branch with all the import dialog work after having requested that it be removed from Damned Lies to not strain the translator is there is a long way to go before we can freeze the strings. Supporting cast Among the number of packages I maintain / update on flathub, LightZone is a digital photo editing application written in Java1. Updating to the latest runtime 25.08 cause it to ignore the HiDPI setting. It will honour GDK_SCALE environment but this isn't set. So I wrote the small command line too gdk-scale to output the value. See gdk-scale on gitlab. And another patch in the wrapper script. HiDPI support remains a mess across the board. Fltk just recently gained support for

it (it's used by a few audio plugins). 1 Don't try this at home.

- Sebastian Wick: SO_PEERPIDFD Gets More Useful (2025/10/10 17:04)
  A while ago I wrote about the limited usefulness of SO_PEERPIDFD. for authenticating sandboxed applications. The core problem was simple: while pidfds gave us a race-free way to identify a process, we still had no standardized way to figure out what that process actually was - which sandbox it ran in, what application it represented, or what permissions it should have. The situation has improved considerably since then. cgroup xattrs Cgroups now support user extended attributes. This feature allows arbitrary metadata to be attached to cgroup inodes using standard xattr calls. We can change flatpak (or snap, or any other container engine) to create a cgroup for application instances it launches, and attach metadata to it using xattrs. This metadata can include the sandboxing engine, application ID, instance ID, and any other information the compositor or D-Bus service might need. Every process belongs to a cgroup, and you can query which cgroup a process belongs to through its pidfd - completely race-free. Standardized Authentication Remember the complexity from the original post? Services had to implement different lookup mechanisms for different sandbox technologies: For flatpak: look in /proc/$PID/root/.flatpak-info For snap: shell out to snap routine portal-info For firejail: no solution … All of this goes away. Now there’s a single path: Accept a connection on a socket Use SO_PEERPIDFD to get a pidfd for the client Query the client’s cgroup using the pidfd Read the cgroup’s user xattrs to get the sandbox metadata This works the same way regardless of which sandbox engine launched the application. A Kernel Feature, Not a systemd One It’s worth emphasizing: cgroups are a Linux kernel feature. They have no dependency on systemd or any other userspace component. Any process can manage cgroups and attach xattrs to them. The process only needs appropriate permissions and is restricted to a subtree determined by the cgroup namespace it is in. This makes the approach universally applicable across different init systems and distributions. To support non-Linux systems, we might even be able to abstract away the cgroup details, by providing a varlink service to register and query running applications. On Linux, this service would use cgroups and xattrs internally. Replacing Socket-Per-App The old approach - creating dedicated wayland, D-Bus, etc. sockets for each app instance and attaching metadata to the service which gets mapped to connections on that socket - can now be retired. The pidfd + cgroup xattr approach is simpler: one standardized lookup path instead of mounting special sockets. It works everywhere: any service can authenticate any client without special socket setup. And it’s more flexible: metadata can be updated after process creation if needed. For compositor and D-Bus service developers, this means you can finally implement proper sandboxed client authentication without needing to understand the internals of every container engine. For sandbox developers, it means you have a standardized way to communicate application identity without implementing custom socket mounting schemes.