Gnome Planet - Latest News

- Christian Hergert: Asynchronous State Machines with Fibers (2026/06/21 10:48)
  Writing state machines gets a bit of a bad reputation because they are often implemented in complex manners which are specific to the problem domain. I think that makes people shy away from writing them when they are truly beneficial, including myself. Where they often go awry is when you have some sort of work that needs to be done asynchronously. This is exceedingly common in UI programming like GTK applications but just as easily found in daemons. Because of this, I see people explicitly avoiding the state machine, or worse, implicitly avoiding its correctness by open-coding a solution across a dozen callbacks. With DexLimiter and DexFiber I find I can write these state machines better. You can use the limiter with a max-concurrency of 1 to get an “asynchronous Mutex” of sorts. No lock management necessary. static void password_daemon_init (PasswordDaemon *daemon) { daemon->limiter = dex_limiter_new (1); } Imagine, if you will, that a limiter is a mutex plus a callback/closure which fires as soon as a slot is free. That means we need a little state to send to our transition callback. /* Define our closure state for a transition */ DEX_DEFINE_CLOSURE_TYPE (StateTransition, state_transition, DEX_DEFINE_CLOSURE_OBJECT (PasswordDaemon, daemon), DEX_DEFINE_CLOSURE_VALUE (PasswordDaemonMode, target)) That is a nice wrapper around defining a struct with a new and free function. Now we can request a transition of the state machine. Since our DexLimiter is an asynchronous mutex (with a single runnable slot), the fiber will not be spawned until it is the highest priority. DexFuture * password_daemon_transition (PasswordDaemon *daemon, PasswordDaemonMode mode) { StateTransition *transition; transition = state_transition_new (); transition->daemon = g_object_ref (daemon); transition->target = mode; return dex_limiter_run (daemon->limiter, NULL, 0, password_daemon_transition_fiber, transition, state_transition_free); } That makes our transition code very clean when you combine the fiber with g_autoptr() and dex_await() to await the completion of futures. So a state machine might look like the following: static DexFuture * password_daemon_transition_fiber (gpointer user_data) { TransitionState *state = user_data; GError *error = NULL; switch (state->target) { case PASSWORD_DAEMON_MODE_HANDOFF: if (state->daemon->mode != PASSWORD_DAEMON_MODE_INITIAL) return invalid_transition (state->daemon->mode, state->target); if (!password_daemon_enter_handoff (self, &error)) return dex_future_new_for_error (&error); break; case PASSWORD_DAEMON_MODE_LOCKED: ... case PASSWORD_DAEMON_MODE_UNLOCKED: ... } return dex_future_new_enum (state->daemon->mode); } static gboolean password_daemon_enter_handoff (PasswordDaemon *daemon, GError **error) { GSocket *control; if (!(control = dex_await_object (create_socket (), error))) return FALSE; ... } What I find nice about this is enter/leave transition components can be customized for the state machine transition. That leaves room for transitions between states which require specialization for correctness. This is much cleaner than ad-hoc callbacks chained together because you can await in the transition fiber for asynchronous work to complete and the state machine itself is preserved. No shoving temporary state in your class instance. No testing hell to see if you caught all the failure cases. No pain with sequencing or order of main loop processing. Hopefully that shows you can use libdex to write more correct and cleaner state machines by keeping the majority of the implementation in one place.
- Hylke Bons: Icon for ChiPass (2026/06/21 00:00)
  Week 23 This week's icon is for whitequark's project: ChiPass: "Store and autofill passwords" Check out all weekly app icons created so far over here and follow my icon creation adventures as they happen (including sketches) on the Fediverse. Need icons? I love designing icons and am happy to contribute them free of charge when your project is Free and Open Source. Funded by community sponsors (every little helps!).

- This Week in GNOME: #254 Commit Graph (2026/06/20 09:42)
  Update on what happened across the GNOME project in the week from June 13 to June 20. Third Party Projects Sjoerd Stendahl says This week I released Lockpicker. Lockpicker is a tool to recover passwords from a hash. Essentially it functions as a front-end for hashcat, making it possible to do password recovery without the hassle of hashcat syntax in a GNOME native graphical interface. It’s mainly useful for anyone that is curious about password recovery, security or as a companion app for CTF-challenges. You can check it out on Flathub. Luiggi R. Cardoso reports Draft’s v2.3 is out now! Now Draft has a new backend and cleaner code, making it easier and faster to add features. This new version brings: Markdown-inspired styling to make it simpler to organize your notes and snippets; Zoom functionality to make the text larger and easier to read; Now the app fully follows your accent color, making the interface seamless with your desktop. Download it on Flathub | Contribute on GitLab | Weblate will be back soon! JumpLink reports Learn 6502 Assembly (learn assembly programming on a virtual retro console) just shipped 0.7.0. The app now speaks Hebrew, fully translated by Menachem (@naattxx), who also tested and contributed the new right-to-left support. That makes Hebrew our first RTL language and prepares the app for more. This release also welcomes a new Indonesian translation by Arif Budiman, alongside the usual round of small fixes and improvements. tanay says Whisp has surpassed 1,000 downloads this week within 15 days(I am in Awe, Thank you so so much). The application now includes several quality-of-life improvements, including the List module, tree-style lists for better organization, note pinning, and keyboard shortcuts for faster navigation. Development is now moving into Phase 2, focused on expanding Whisp beyond note-taking. Planned features include a built-in Math Engine for calculations, unit and measurement conversions, and Image-to-Text functionality powered by OCR. Thank you to everyone who has tried, reported issues, and contributed feedback. Community support has been invaluable in helping shape the project. Project website: https://tanaybhomia.github.io/Whisp/ Source code: https://github.com/tanaybhomia/Whisp If you’d like to support development, UPI donations are now available through the website, and COFI support is planned soon. Gitte ↗ A simple Git GUI for GNOME Christian reports Gitte, a simple Git client for GNOME built with GTK4, libadwaita and Relm4, just got its 0.7.0 release! The headline feature this time is a visual commit graph in the commit log: branches are drawn with color-coded lines, so commits off the mainline get a distinct per-branch color and it’s much easier to follow how history actually branches and merges. Stashes and pending changes show up as commits in that graph too, giving you a complete picture of your repository at a glance. You can also drop multiple commits at once now. Diffs got a lot smarter as well. Gitte now detects renames, and the word-based diff has been overhauled to highlight intra-line changes more often. There’s also a new shortcut (Ctrl+D) to switch between staged and unstaged in the working copy view. There’s more control over your lists and directories: you can toggle whether Gitte recurses into untracked directories, and the branch and tag lists now have a configurable sort order, with global defaults available in the preferences. This release also comes with a broad UI overhaul. The main layout has been greatly reworked (special thanks to Brage Fuglseth!), and the preferences dialog is now a standard Adwaita preferences dialog. The columns now each have a distinct background color, the sidebar was reorganized, and the commit-graph and “rebase in progress” banners were revamped. On top of that come nice new action buttons in the sidebar, better spinners if an operations takes a bit longer, pane widths kept in sync across all views, and the usual round of consistent title casing and spacing fixes. Under the hood there’s a performance pass too: file diff lists now show at most 100 files (30 expanded by default, with an option to force loading everything), more operations run asynchronously, and the diff view isn’t reloaded when it doesn’t need to be. And of course there’s the usual pile of fixes: diff lines are redrawn on theme switch, scroll position is preserved while staging and unstaging, long labels break correctly, tag badges got a better color, and the window size is now saved correctly — including when quitting with Ctrl+Q. Get it on Flathub, for macOS or have a look at the Code. GDM Settings ↗ Customize your login screen. Mazhar Hussain says

It has been almost 2 years since I last posted about GDM Settings on TWIG. While there have been some releases during that time, the development wasn't happening really consistently. I'm really sad to announce that the development of GDM Settings is on hold for now. I'm not sure when (or if) I'll pick up the development of this app again. See issue#324 for information. Thank you all for the support! It really meant a lot. ❤ Shell Extensions Christian W reports macOS-style text capture now also for GNOME: select any screen area, OCR it, copy instantly, optionally translate with this new extension https://extensions.gnome.org/extension/10209/snap-text-extractor/ Tomáš Gažovič says RSS Feed extension is back after a long time and ported to latest GNOME. Same old feeds, but with a modern look. If you like your news directly from shell you can give it a try https://extensions.gnome.org/extension/948/rss-feed Project page: https://github.com/todevelopers/gnome-shell-extension-rss-feed Miscellaneous balooii reports For the two people out there that might be interested in trying out GIMP 0.54 from 1996 on your modern Linux system: Now you can, I created a Flatpak just for fun. It's the last version using Motif before it's creators decided to kick off a little toolkit called GTK... https://gitlab.gnome.org/balooii/gimp-0.54 GNOME OS ↗ The GNOME operating system, development and testing platform Bilal Elmoussaoui reports GNOME OS is now using oo7 by default as a replacement of gnome-keyring-daemon, oo7-portal for the XDG Secrets portal implementation and oo7-cli as a replacement for secret-tool. 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!

- Hari Rana: Draft-Driven Blogging (2026/06/20 00:00)
  From 2021 to 2023, I was really motivated to write articles regularly, but that is no longer the case. Most of my energy goes into programming nowadays. Whenever I try to write about a complicated topic in a digestible manner, I quickly lose motivation and don't publish it. For half a year, I've been trying to write an article about the thought process that went through when making the month view in GNOME Calendar accessible, as well as the implementation details. However, explaining complicated technical details into something that is simultaneously digestible to non-developers interested in accessibility and free and open-source calendar application developers requires me to withdraw my knowledge and assumptions, consider the perspective of someone who is not knowledgeable in this topic, and then recall the events that led me to take certain decisions, which demands a lot of energy. Due to a lack of motivation, I want to try a different approach. I am calling this approach "draft-driven blogging". Instead of publishing articles once they are complete, I will publish the draft publicly. This draft may contain keywords, incomplete sentences, random notes, empty sections and other characteristics that are only found in drafts. I will then iteratively improve the draft until it is considered finished. This approach makes sense to me in terms of publishing and getting things done. I tend to seek perfection, which is great for maximizing quality, but it comes at the cost of motivation. Without external pressure, I am not motivated to fix something if it is not already publicly available. Seeing an unfinished blog post publicly is simply appalling. As it's ugly, it motivates me to fix and complete it. So, instead of writing and publishing the 'perfect' article, I publish the ugly draft and complete it out of spite. Maybe "spite-driven blogging" is a better term for it? Of course, communication is important. With drafts, I will add a disclaimer stating that the article in question is a draft. The published date will be the date of the last edit, and all previous drafts will be deleted. To avoid spamming RSS feeds and notifications, I will try to republish drafts infrequently. It's all an experiment; it might work well, or it might not. I might keep this approach or just pretend that I never tried it in the first place. We'll see.
- Hylke Bons: Icon for Hex Colordle (2026/06/19 00:00)
  Week 22 This week's icon is for Krafting's project: Hex Colordle: "Find the color" Check out all weekly app icons created so far over here and follow my icon creation adventures as they happen (including sketches) on the Fediverse. Need icons? I love designing icons and am happy to

contribute them free of charge when your project is Free and Open Source. Funded by community sponsors (every little helps!).

- Sam Thursfield: Status update, 17th June 2026 (2026/06/18 07:58)
  This month I'm mostly listening to music by Nu Genea, Danalogue and Noon Garden. I'm going to tell you about a big change I'm proposing for folks using Freedesktop SDK to build operating systems. And I'm also going to talk a bit about the GNOME Foundation elections. Maybe I'll do that first. GNOME Foundation elections The GNOME Foundation is a democratically organised not-for-profit that grew from the GNOME open source project around the year 2000. Anyone who contributes to the GNOME open source project can be a member of the Foundation, which allows (among other things) periodically voting in a Board of Directors who govern the Foundation. You probably know all of this. Back in 2001 the Foundation was a lively active space. Check out the election results from 2001: 11 candidates were selected out of 25 candidates, some of whom made pretty wild campaign promises such as banning all mentions of proprietary software. When I became more involved in GNOME, ten or fifteen years ago, the Foundation seemed pretty boring and sensible and not many people volunteered to be directors. Here's a mail from 2017 of someone complaining about not enough candidates and low voter turnout. This year's election has nine eight candidates for five seats, and a lively debate. Two years ago a big explosion happened in the community, and we are still dealing with the fallout and, in many cases, still piecing together what actually happened. (It seems like the explosion had been building for a long time and maybe the boring and sensible days of 2017 weren't as boring and sensible as they appeared from outside.) I am not impressed by the tone of some of the discourse, everyone involved in the campaign believes in what they are doing and deserves respect, but it does make me optimistic about the future of the GNOME Foundation. Questioning things is healthier than ignoring them. My quiet theory is that the dynamics of open source have changed fundamentally now that LLMs are a mainstream technology. Code is less of an asset than it ever was. A lot of work in the desktop space since 1997 has been simply keeping pace with the rest of the industry: Apple introduced glassy window bars and so we had to have them in GNOME too, Apple introduced "retina" displays and now we need fractional scaling, Apple introduced the App Store and now we need Flatpak, and so on. All of these are huge engineering efforts requiring a lot of new code. Now the industry is out of ideas. Apple this year are announcing AI integration and more glassy window bars. So if code is not the asset, what is? The people, as it always ways. And in an increasingly hostile and untrustworthy internet, where you can't even trust websites any more, a resilient autonomous and trusted structure like the GNOME Foundation, with a battle-tested democratic structure, and strong moderation capabilities to keep out the increasingly automated and vociferous trolls, is a very valuable thing indeed. (No wonder the trolls see it as a threat). It's hard to imagine a parallel universe where there's no KDE eV and no GNOME Foundation, but I suspect we would miss both of them. Clearly all of the candidates believe in the Foundation enough to run for election. Remember that it's an unpaid position with a lot of responsibility and minimal benefits. Being a director is a personal sacrifice. So thanks to everyone who keeps it working. freedesktop-sdk.bst:public-stacks/runtime-gnu.bst Onto the more technical material, I guess. The Freedesktop SDK is a widely used Flatpak app runtime that powers thousands of apps on Flathub. You probably know that, too. Flatpak aimed from the beginning to be distro-independent, and consequently the Freedesktop SDK isn't a repackaging of Debian or Fedora or Alpine Linux, but something more like a DIY Linux From Scratch build. As an app user you don't notice any of this, because it's very well executed and apps just work. Again, it's hard now to imagine a parallel universe where the main Flatpak runtime was Fedora in a trenchcoat, but perhaps that would have impeded the success of Flatpak. (Of course Canonical still built their own app store technology, but I suspect that Canonical re-inventing things is part of every parallel universe). So Freedesktop SDK has build instructions for common Linux tools, utilities and libraries, and they are so good that most BuildStream projects end up junctioning Freedesktop SDK to reuse them. (I covered this in more detail back in April). In theory this brings a virtuous cycle: we

use FDSDK in industry and that funds maintenance and improvement of the build instructions, which in turn benefits the Flatpak runtime which doesn’t have any source of funding of its own. I’ve been working on a slightly tricky intersection between these two worlds, which I call “Choose your own userland”. It makes a relatively small change to a stack element in Freedesktop SDK, but one which has big consequences for BuildStream projects that junction the project. (And no immediate consequences for Flatpak users, but you could see it as future-proofing). A stack element is a group of elements. Freedesktop SDK provides various “public stacks” with useful element groupings. Most of these are related to build systems, like public-stacks/buildsystem-autotools.bst which includes everything you need to run builds with the crusty old GNU Autotools build system. Then there’s this special one: public-stacks/runtime-minimal.bst, which as of today includes the following: Root filesystem symlinks C/C++ platform libraries like GNU GLIBC, The GNU Bash shell (and its dependencies Readline and ncurses) GNU Coreutils, and all their dependencies This stack is a recent addition, announced in the release notes of last year’s FDSDK 25.08 major release: [BREAKING CHANGE] It’s now possible to create more minimal runtimes thanks to rework of bootstrap-import.bst. This adds a new stack public-stacks/buildsystem-make.bst which is essentially same as the original bootstrap-import.bst. There is also a new stack public-stacks/runtime-minimal.bst that is intended to provide a minimal environment that you can shell into. More info in the related issues: #1728 (closed), #1523 My selfish motivation for this change is I want to build embedded systems that don’t include GNU Bash and Coreutils at all, using BusyBox to provide the shell and utilities instead. This is hard today with FDSDK because every element unconditionally depends on Bash and Coreutils, so how can I remove them from my final system? But coincidentally, in the desktop world we are also seeing GNU Coreutils replaced with uutils/coreutils, a reimplementation in Rust which is already the default in Ubuntu since 25.10. So there’s another reason we might not want to hardcode a specific coreutils implementation in the lowest level stack. The idea implemented in my branch came from Abderrahim and is delightfully simple: just drop Bash and Coreutils from the runtime-minimal stack, and have elements opt into them explictly. On hearing the idea, I wasn’t sure how this would work, so of course I was effectively nerd-sniped into trying it. The result is as we’d hoped, it allows you to build systems with alternative coreutils. The FDSDK includes some example VMs, and here’s an example of one of them booting with uutils/coreutils (taken from MR!31779): https://gitlab.com/-/project/4339844/uploads/04ea5b38ae78b5ee6ed175ea9ea54369/FDSDK_example_system_with_uutils-coreutils.webm So the approach works. My main concern was the amount of churn we would cause if we change the meaning of runtime-minimal.bst. Of course, we often still want GNU Bash and GNU Coreutils, so my branch adds an additional public-stacks/runtime-gnu.bst element that brings in a GNU userland. I added Bash and Coreutils into all the public-stacks/buildsystem-*.bst elements too as we still want them at build time. That means that for most elements the change is actually transparent. You just need to ensure that each output explicitly includes a shell and utilities of your choice as runtime-depensd. To test things further I tested the changes in branch of gnome-build-meta. It was pretty boring working through various build failures to get to a new dependency graph, but I came out the other side still convinced that this change is a good idea. (You can see my gnome-build-meta branch here, bearing in mind half the changes are actually dealing with differences between FDSDK 25.08 and ‘master’). There was some lively discussion on the MR and I’m still not entirely clear if this change is going to land, much as I like it. One sticking point is a fear of landing big changes and not having enough people to deal with the fallout, and as an open source maintainer I certainly know that feeling, so I have more testing planned still. Another complaint is that this change reneges on the promise from 25.08 about public-stacks/runtime-minimal.bst, that it “is intended to provide a minimal environment that you can shell into.“. You can’t shell into anything if there’s no shell, of course, so I can’t argue with the premise. But I am missing why this is a big deal. I’ve always had a bad time in BuildStream build shells because I just want to edit a file for testing and dammit there’s no Vim or even Vi or even Nano… in fact we don’t even seem to have less?!

So I’ve always wanted a way to overlay elements with debug tools in my shell, and it turns out that “bst shell should be able to stage the specified element on top of a base element” is a feature request that’s been open since 2018. If you use FDSDK as a junction and you like the idea then I’d appreciate comments on the MR. (If you hate the idea, I’m sure you’ve already switched tabs and are half way through posting an irate comment ;-). I am of course prepared for an outcome where this change doesn’t land, and it may indeed lead to some separation of “Linux OS & compiler bootstrap using BuildStream” and “Base Flatpak runtime” into different projects. My gut feeling is that this would be a bit like trying to carve a single grape into two pieces, i.e. there are still few enough people who actually want to maintain build instructions that it makes more sense to collaborate in the same repo. Thanks as always for reading!

- Michael Calabrese: Pitivi Timeline Ruler | Standalone Beta Progress (2026/06/18 00:00)
  Hello GNOME, This is a progress report on the Pitivi Timeline Ruler Rust rewrite. Progress We are rewriting the Pitivi Ruler in Rust and gtk4 snapshot logic to improve performance and memory safety. At its current stage the ruler is being constructed as a standalone widget in a personal repo that can be found here: Pitivi Timeline Ruler GTK_DEBUG=interactive cargo run --example sandbox Any feedback on the code is greatly appreciated! Structure I am structuring this around rendering a single interval of ticks between two major ticks once, then stamping that across the duration of the project, as seen in draw_single_interval(). With the ticks stamped across the timeline I am then rendering a cache of Pango labels for the timestamps that are stamped across the visible window. I made the decision to use a BTreeMap for the cache for ease of iterating chronologically through the stamps and for dropping out-of-bounds keys. The logic for this cache handling primarily lives in update_label_cache(). After some feedback from members of the video editing community, I made the decision that minor ticks should always be clean multiples of frames in the time period. This logic is implemented in calculate_minor_divisions(). Next Steps The primary goal I am focusing on next is implementing the gesture handling, including click and drag actions. Once gestures are implemented I am going to begin moving a lot of the math to traits so that I can write a mock_env and a live_env to start writing some unit testing.
- Jakub Steiner: Things That Last (2026/06/17 00:00)
  One of the great annual trips we do with a bunch of friends is a train trip to Jakuszyce, a tiny stop in neighbouring Poland, and ride along the contour line through one of the most beautiful places in the Jizera Mountains. There's only one proper climb from Smedava to Knajpa, the rest is fast. A joyride. Catching up on our lives on the train and a joyride home is the best combo. I tend to think of myself as a friend of repairs, of making things last. I have sadly had to retire our washing machine after a good 25 years. The dishwasher before that served us more than 15. A boiler and heater had to go this year after about 20. I had my previous car for 13 years and felt like I was bailing too soon, even though there were quite a few issues with it at the end. None of those things ever felt new to me by the end. They most certainly showed their age. But the bike. The bike I ride every year on that trip, the one leaning against the wall in the shed right now — it still feels like my "new bike". I replaced the tires and brake pads last year and the thing screams. It is such a joy to ride. It feels current, alive, like something I just picked out. Until a friend sent me a photo his Google Photos app reminded him of. A very young version of myself is sitting on that exact bike. Fifteen years ago. Nothing has aged except me. Bikes just don't age like we do.
- Jussi Pakkanen: Beware of Star Trek managers, especially when bearing MBAs (2026/06/15 19:29)
  Almost exactly three years ago the Oceangate submarine implosion happened. The disaster came about when a billionaire called Stockton Rush created his own unclassified submarine to go sightseeing on the Titanic. Ignoring all advice from experts he created a "macgyveresque death trap" that eventually killed him and sadly also 4 innocent people. The whole thing was a massive display of stupidity and arrogance with

unfortunate outcomes. We are not going to go into the actual event any deeper, but those interested can find lots of material online.Instead we are going to look more deeply into one often overlooked points of Stockton Rush's character. Apparently he felt like he was something of a "new James T. Kirk" (link1 paywalled, link2). Liking Star Trek is not that unusual. I'm guessing that more than 99% of the readers of this blog are fellow Star Trek fans. The problem lies elsewhere, but to understand it we first have travel back in time.A brief overview of the British navy during the Napoleonic wars (by a non-historian, so probably inaccurate)The original concept for Star Trek was, approximately, The Adventures of Horatio Hornblower in Space! The Enterprise is basically a British warship sailing through the vast ocean of outer space. The command structure mirrors this, where you have a captain, navigator, ship's doctor and so on. The Next Generation leaned into this even further by having a first officer and so on. The original Star Trek never went into detail on how the main cast got to their current positions, just that there was an Starfleet Academy they went to.In the Napoleonic era of Hornblower things were quite different. Anyone who wanted to become a captain pretty much had to be from the upper classes. They had to obtain a letter of recommendation so that they could join a vessel as a midshipman at the age of 13 or so. They were expected to be able seamen by this time and then spent the next six to seven years working on the ship rigging sails and doing all manner of random jobs. This went on for six to nine years depending on circumstances, after which the person could take a formal examination to become a lieutenant. The test was not trivial, many people could not pass even after trying multiple times.A lieutenant then had to work successfully for several years before obtaining the rank of captain. Even that did not guarantee a commission. Some captains never commanded a ship simply because there were not enough of them to go around. All in all becoming a ship's captain was a long and difficult journey. In a surprisingly non-British turn of events it was not possible for aristocrats to sneak past the gates. Getting a midshipman position was obviously easier with connections, but the lieutenant's test was something they had to pass on their own.All of this is to say that every captain of the time was an expert with decades of working experience on many different positions aboard the ship.What does a captain actually do?[Note: I have not fact checked this portion at all. Feel free to consider it fanfiction.]The year is 1808 and we are aboard a British warship about to leave for a mission of great importance. The captain gives the order to set sail. Whistles are blown, bells are rung and sailors springs into action. Every single man, with one exception, is either doing manual labour or directly supervising their underlings. That exception is the captain, who seemingly stands around doing nothing (at least if you ask the crew). This is not so.What he is doing is crucial. He is observing the state of the ship and her crew. This includes things like overall crew morale, any aberrations from normal operations that could cause problems, thinking of workflow improvements and so on. In a sense he has to sense the ship itself. This only works because of two things. First of all he has personal experience doing the exact work he is observing. If you have not personally "been there", you can't really know if a crew is working well or not. You need a "gut feeling" to be able to sense this. Secondly the captain does not have any manual labour so he can focus all of his mental energy on observing the ship's state. He is preparing for all the unexpected things that may occur in the future. This can only happen if your brain is free from menial tasks.This is exactly what most books on business and project management advocate. It is a time tested way of improving your chances of success. A highly skilled commander can take an average team of people and lead them to victory. It is the basic plot of most military and sports movies.Getting back to the presentNow take a typical modern day billionaire-via-inheritance and show them Star Trek at an impressionable age. Do they see the advantages of education, hard work and ethics? The foundation upon which Gene Roddenberry carefully built the show? Hell no! What they see is this (TNG screenshot used because TOS did not have a suitable maritime episode).And then they think: "Wow! I want to be exactly like that! Parading around in a funny hat while everyone obeys my orders without question is my life's mission from now on. And I get to have sexy space sex with hot sexy space ladies of sex whenever I want. This appeals to me even more profoundly than Atlas

Shrugged." Some of them might go on to watch Master and Commander and shed tears upon realizing that they cant publicly flog employees for failing to salute their superiors. Yet. Expect this to be made legal in Silicon Valley any day now.Liking Kirk is not in any way a bad thing. Wanting to "be just like Kirk" is, because in the real world running a business like Kirk runs the Enterprise is a terrible way to do things. An example will illustrate this nicely. Let's imagine a random episode where the Enterprise has gotten into trouble. Eventually Kirk will call for Scotty and tell him: "You need to <babble> <babble> <babble>." Scotty will then reply with a varying level of scottish accent something like: "I cannae change the laws of physics, captain". Kirk will then say the same thing again, just more aggressively and in a close up shot. Scotty replies with "Well in that case I can get it done in sixty minutes." Kirk counters with: "You have five." And thus the problem is solved. Kirk gets a commendation for incredible valour under stress while Scotty, who did all of the actual work, is never mentioned.In "Kirk style" management the Big Boss tells his underlings what to do. If they try to give any sort of feedback, the Boss ignores everything and just repeats his original orders again and again until the other party yields. The only reason underlings ever resist The Vision is that they are lazy and it is the job of the manager to put them in their place. This seems like a bit from a comedy show, but I have unfortunately worked under bosses like this. Either you try to talk at least some sense into them, fail, get labeled as a "not team player", watch the project crash and get blamed for the failure, or you try to do the impossible task given to you, fail watch the project crash and get blamed for the failure.Another major problem with Kirk is that due to the way tv shows and movies need to be structured, he is actually an obsessive gambler. The stakes must always get higher and the ways to get out of trouble must become ever crazier. Kirk will break any and all laws and regulations he sees fit and then, once he has succeeded, no disciplinary action is taken. The ends justify the means. Idolising this sort of behaviour leads to thinking that wild one-in-a-million gambits will succeed at least 99 times out of a hundred. And even if it fails, you can get out of it by betting everything on an even bigger gamble. The real world does not work like that. Reality is not a story and you are not its hero. It does not owe you eternal, or even eventual, success. Had Stockton Rush survived his death trap, he would most likely have faced criminal charges and, if convicted, gone to jail.The myth of the existence of the professional managerLet's make one last detour in the 1800s and assume that the 7th Earl of Sidcup or some such really wants to get his idiot son instated as a captain. He and contacts the appropriate naval officers."My offspring needs to become a captain of a ship post haste!""Well first he has to become a midshipman and ...""Phah! None of that nonsense. It's way too slow and not becoming of my statute. Also my son is 35 years old so the post of a midshipman would be beneath him.""I see. Well what sort of prior naval experience does he have?""None.""Has he even ever been out to sea?""Not to my knowledge. But that does not matter. He is highly skilled in using the abacus excelius to compute annual budgets.""By himself?""Of course not. That is what secretaries are for. He just gives them orders. That he can do. And that is all that matters. Same as in sailing."This person is unlikely to get his wish with this line of reasoning. On the other hand in modern business life this is common. For example when startups get VC money, a common requirement is that they need to get a "proper manager" as a CEO. Typically this means the investor's friend, and more often than not an MBA.Contrary to common belief, having an MBA does not make you incompetent at managing a technical company (though there is a strong positive correlation). It is entirely possible to be a good manager on a field you have no personal experience in. You just have to have a lot of humility, listen (actually, properly listen) to your employees and let the people with hands-on experience make the most technical, product and development decisions. In other words you have to be the enabler, not the maverick decision maker. People with these sorts of personality traits are rare and typically their career choices steer as far away from getting an MBA as possible.The absolute worst thing happens if the CEO in question combines the (lack of) skills of an MBA with the attitude of Kirk. That leads to incompetent decisions based on willful ignorance, executed with the fury of an egomaniac who refuses to even entertain the notion that they might be wrong. Further, any person inside the

organisation who dares to point out potential flaws in the plan will soon find themselves outside said organisation. Disagreement is treason. Treason shall not go unpunished.In the 1800s the British navy could be said to be the best in the world. It seems plausible that one component of this success was the requirement that the officers running their ships had to have actual experience operating the ship. Not looking at other people operating it. Not pretending to read about operating it for a test. Actually doing it. If we look around how MBA wielding sociopath CEOs are enshittifying absolutely everything about the tech industry, bringing this requirement back into active use starts to feel awfully tempting.Epilogue: Why doesn't everything immediately explode?A reasonable counterpoint to everything written above would be that if managers truly are that bad, shouldn't all those companies be bankrupt by now? In an ideal world they would be, but there are opposing forces that keep them going.The first one is that all corporations have inertia. If you took an established major company and actively started to mismanage it to death, it would still take years for things to eventually collapse. The second one is a dirty little secret. Many employees care more about the product they work on than "corporate visions" that seem to stem from overuse of peyote. They don't blindly obey idiotic commands but instead try to make things silently work within the system. Basically this means that corporations thrive despite their mad kings, not because of them. I know several people who have worked in these kinds of organizations and this is not as rare of an occurrence as one might imagine.I have also experienced it personally. Years ago I was at a company, whose CEO (who, to the best of my knowledge, did not have an MBA) wanted to change the company's product so that it would do a specific thing X. Everybody thought this was a horrible idea and tried to reason with him using solid business and technical reasons (which turned out to be 100% correct). That failed. Spectacularly. This lead to an eternal series of secret meetings. The participants were main developers and all managers except the CEO. The only item on the agenda was "How can we make the CEO think that we did what he ordered while doing the exact opposite".Eventually we did succeed, but boy was that a surreal couple of weeks.

- Arun Raghavan: Notes from the PipeWire Hackfest 2026: Part 2 (2026/06/15 02:23)
  (these notes are being posted in two parts to make the length more manageable, part 1 is here) Continuing from where we left off, about topics discussed at the PipeWire hackfest in Nice… DSP features We discussed a number of features related to digital signal processing blocks which are typically realised on specialised hardware (often a DSP core that can directly interface with physical audio inputs and outputs on your laptop/phone/…). There is currently no standard way for the firmware running on these DSPs to signal what features can be realised directly on DSP. We also would want to allow such features, if exposed from PipeWire, to be realisable on CPU. Now we do have a way to hide away signal processing in a specific node, which is the filter-graph parameter on the audioconvert node that wraps all audio nodes. We could extend this mechanism to allow the internal node (say the ALSA node implementation), to expose what filtering it can perform “in hardware” (i.e. the software running on DSP). This would allow the audioconvert to delegate some or all processing to the internal node, with fallbacks available on the CPU. We would need a number of pieces to do this, including: Some standard definition of filters and associated parameters, so different implementations could have a standard “API” to express any given filter. The DSP block would need to expose what features it has and how they might be used. We could imagine extending the ALSA UCM configuration to do that. The audioconvert node would need to have a way to push down filter-graph params to the internal node, and negotiate what work it is doing vs. what is being delegated This is a non-trivial effort, but gives us some sketch of what might be possible. More DSP features In addition to standard filters, we spoke about two topics that have come up commonly in the past. The first is some way to expose the processing graph in the DSP, so PipeWire and other userspace daemons have a better view of what is happening on the DSP. With the ability to push dynamic topologies to DSP, there was some renewed interest in exposing and

using the ASoC DAPM widget graph. As always, the devil is in the details. The second thing that came up is speaker calibration. There is a lot of processing and tuning that goes into driving speakers on modern devices as much as possible without destroying them. Some of these are one-time parameters decided at product design time, and some of these translate to runtime parameters based on voltage and current feedback from the speaker amplifier. For some systems (like Qualcomm platforms), speaker calibration might be run on each system start to perform dynamic tuning. We had some discussion of how this might tie in with the rest of the system for both determining the parameters (separate startup daemon vs. in-process initialisation), as well as uploading parameters to the speaker (some ALSA UCM extensions to load parameters on PCM open but before start, or preloading parameters into ALSA kernel controls and having the driver feed them in at the right point). Volume limits A way to set a limit on the maximum volume for a given device has been a common user request ([1] [2]). We discussed the possibility of creating a per-route property (with a fallback to the node, if there are no routes), which WirePlumber could manage to provide users a simple interface to control. Since the hackfest, Wim has already done some work on this, and we need to bubble this up as a more user-accessible setting. Performance A number of performance-related topics were discussed. The first was an option of a combined DSP mode, where instead of one port per channel, a node would expose one port for all the channels of the stream (but continue to run in the configured “DSP” format/rate). This would improve stream performance for non-JACK-like use-cases, especially in resource-constrained environments. On the WirePlumber side, there was a discussion about using LuaJIT instead of standard Lua. There are some compatibility issues to be determined there (such as language version supported, etc.), but there might be some quick performance wins to be made if this is feasible. There is a plan to move some of the WirePlumber core to Rust, and that might be a good time to also port over some of the more standard functionality that tends not to change from Lua to Rust (though that could happen in a Lua->C transition and does not really need to wait on a Rust port). Declarative Session Management Another interesting, and broader, thread is the imperative nature of WirePlumber scripts – that is, policy decisions and associated action are often interwoven. It might be helpful to be able to make a clearer split where all policy decisions are first run, and then decisions are translated into actions at one go. There are some historical choices that make this hard – for example, changing the profile of a device might create and destroy nodes, which makes it hard to be able to make decisions that are independent of the action. There were some ideas around redoing the profile concept such that all nodes are always exposed, but nodes could get a new state to signal availability (and profiles that would allow availability to change). That might make a declarative system possible to implement. We also discussed the possibility of a “transaction” system. Something that would allow a client to submit a set of objects (think links between nodes), and then “commit” that transaction. This would also help reduce the number of roundtrips between PipeWire and WirePlumber, and generally help performance. Bluetooth Being colocated with the BlueZ face-to-face meeting, we had representation from the BlueZ community, so we were able to dive into a number of topics related to Bluetooth, primarily LE Audio. The first topic was Auracast, the LE Audio system for broadcast audio, allowing listeners to tune into public broadcasts in a space, or to have a device stream audio to multiple headsets concurrently for shared listening. George had a demo system showing an implementation of Auracast with PipeWire, WirePlumber and BlueZ. We had some discussion of where this feature should live, and the consensus was that we would probably want a separate daemon to manage Auracast settings and loading up the appropriate nodes (either for receiving or sending) based on users’ preferences. This led to a more general discussion about the current split of the Bluetooth implementation in PipeWire being SPA modules, which include streaming and some policy, and a lot more policy living inside WirePlumber. We could, and likely should, move all of this into higher level PipeWire modules instead, which could make these easier to work with overall. There was also a discussion about the complexities of LE Audio, and the state of the current user experience with actual devices: Device interop is not

always great, as the spec is new, the BlueZ implementation is still being completed, and device implementations seem of variable quality Reliable pairing/feature detection is hard, partly due to how BlueZ exposes the ability to talk to devices in Bluetooth Classic or Bluetooth LE modes Pairing left/right pairs currently needs individual pairing, which does not seem to be needed by other implementations (Android for example) Inter-device synchronisation might need some work as well While there is much work to be done here, the pieces are coming together for first-class LE Audio support on Linux-based systems. Audio analytics We also spoke about “analytics” – using local neural networks to implement things like text-to-speech, speech-to-text, language translation, or other forms of processing. These pose an interesting problem, because they look like a standard-ish audio stream on one side, but are effectively a sparse stream on the other side if we are talking about text. Even conversion between languages does not look like a standard filter, because the underlying model might consume a varying amount of data before generating an output, and the input and output lengths are not tightly correlated. While it should be possible to implement such a system with PipeWire, it is not quite clear whether we should. As the application space in this area becomes more mature, it may become clearer what the right place in the stack is for these features. Click detection and elimination We spoke about detecting and eliminating clicks at the stop or start of a stream. If an application is playing back audio, and suddenly stops (i.e. feeds silence, or just nothing), then the sudden drop in the signal might cause a click to be output. If you think of the corresponding waveform as representing the physical displacement of the speaker, then the drop to zero is like a sudden brake to a halt, which isn’t possible, and manifests as a jolt that you hear as a clicky noise. The same analogy holds for resuming from a pause, but in the opposite direction. The solution is usually to smooth out the end of the sound by fading out, but most applications do not do this, so this problem manifests quite clearly for most browser or application streams if you listen closely. Wim described a number of experiments he has done for detecting such abrupt changes in audioconvert, but he was not happy with the results. We discussed some of these approaches, and what might work as acceptable tradeoffs to capture the most common cases while still trying to respect the integrity of the signal being sent by the application. (sorry about the vagueness here, I missed taking more detailed notes) Miscellanea The rest of the discussion covered disparate topics that I don’t have long form notes on: Hardware profiles: Shipping hardware-specific configuration for PipeWire and WirePlumber is hard. We discussed some approaches using context properties and conditions, but this is an area that needs more work. Data loop management: PipeWire allows splitting work across data loops so different nodes in a graph can be assigned to different threads. This is currently an all-or-nothing system, where either all nodes go to a single data loop, or every node must be manually assigned a specific data loop. There was some desire to have the ability for there to be a default data loop to make the manual management less cumbersome. ACP -> UCM: PipeWire inherits the ALSA card profile configuration from PulseAudio, which has been helpful in making the migration path smoother on most hardware. There was always some desire to have a single configuration system (probably ALSA UCM) for all hardware, but this likely needs some work on what we can express in UCM configuration, but we also need to clean up how we translate our UCM handling code (George has an RFC for this). Thanks That’s it, thank you for reading if you made it this far, and a shout out to George, Mark, and others organising the event! It was great to see continued interest and so much exciting work that is yet to come. I hope to see more of the community in the next edition of the hackfest.

- Arun Raghavan: Notes from the PipeWire Hackfest 2026: Part 1 (2026/06/15 01:07)
  (these notes are being posted in two parts to make the length more manageable, part 2 is here) The PipeWire community organised a hackfest in Nice, France, colocated with Embedded Recipes, the GStreamer hackfest, and a number of other events. In attendance were members of the upstream community, as well as folks interested in PipeWire from Collabora, Red Hat, Qualcomm, Stream Unlimited, Texas Instruments, and

Valve. In some cases these were the same person wearing upstream and professional hats, as some of us often do! :) It was two days of fruitful and deep technical discussions, and lovely evenings hanging out in the Côte d'Azur. Shout out to George Kiagiadakis and Mark Filion for putting this together! Beautiful view of the Côte d'Azur The topics were disparate and can be somewhat esoteric for folks who are not familiar with the Linux audio space. I will try to strike a balance between providing context and summarising the finer details we discussed. Please feel free to write in if I missed or can expand on anything. Multistream nodes A recurring topic for the last couple of years has been supporting multistream nodes. The PipeWire API currently offers a pw_stream interface that can offer a node with single input or output (closer to the PulseAudio API), and the pw_filter interface that provides a lower-level freeform API to individually manage ports on a node (closer to the JACK API). The stream API while convenient, can be a bit unwieldy for realising concepts such as loopbacks and filters, because each set of inputs and outputs needs to be implemented as an individual node. If you've ever loaded the loopback module, for example, you would have noticed that there are two nodes created for each instance. Wim has created a version of the API that allows a node to provide multiple streams, which allows us to keep the conveniences of the stream API, but more easily express ideas like the loopbacks, filters, etc. Each stream is effectively a group of ports on the node, and nodes can have an arbitrary number of input and output streams. The code on the PipeWire side is ready. The primary idea is there will be a PortConfig param per stream, and this is where the format of the stream, and other metadata expressed on port groups (which is essentially what a stream is) will live. We discussed what is needed in WirePlumber to make sure the linking logic adapts to this concept, and Julian will be implementing that in the coming weeks. Settings PipeWire has a generic metadata system based on the JACK API that is used for storing metadata (allowing you to attach a key/type/value, optionally attached to an object). This is also used by WirePlumber to provide its settings system (see wpctl settings), along with some key features such as a schema and persistence. We discussed that it might be nicer to have the concept of settings as a first-class citizen, and possibly even standardise some settings for desktop wide usage (such as common processing elements). There was consensus that: A new settings interface (instead of extending metadata) would make sense The API should be asynchronous, and can fail A schema for valid settings and their types could be exposed as a well-known metadata key Implementors of the interface would perform validation Security We spoke about the current state of security for applications using PipeWire. For context, PipeWire has a fine-grained permissions model where each client can have selective access to what objects are visible to it, and what actions it may perform. There is also a less granular system, where a "manager" application can connect to the manager socket for full access. We broadly think about restricted security for sandboxed applications (primarily Flatpak). One scenario is sandboxed PulseAudio applications getting full access via the pipewire-pulse server on the host. The discussion on this concluded that there is a way for pipewire-pulse to forward enough security-related information from sandboxed applications for us to apply sandbox restrictions to them, and we need to make that system work. There was a discussion that it might be reasonable for our default policies to apply for all applications connecting to the regular PipeWire socket to be restricted (this does not prevent malicious applications from accessing the manager socket, but helps applications not do bad things erroneously). This might be disruptive to introduce as a default change, so we might implement it via an opt-in setting so that there can be some broader testing and refinement of default permissions before flipping the switch for all users. There are a number of mechanisms related to how security context properties are relayed, and how those properties are used by WirePlumber to determine permissions. We need to document and verify the expected behaviour here. Flatpak and Portals Relatedly there was a discussion about how things should fit in with Flatpak, and Sebastian Wick from the Flatpak team joined us briefly on the second day. There was some discussion of making sure the PulseAudio socket is provided to the sandbox in a similar way to the PipeWire socket, such that some additional security properties can be assigned from the host in a

way that the sandboxed client cannot override. We agreed that we needed the ability for applications to specify with some granularity what permissions they require (via portals), and for us to grant only that (with user intervention, if needed). Broadly this is: Playback (optionally enumeration of sinks) Capture (optionally enumeration of sources) Default visibility of only the application's own nodes We also spoke about how we might want to associate PipeWire objects with applications. With Flatpak moving to using a cgroup for each application, this should become easier. We may also want to be able to have a way to associate a stream with a specific window (to, for example, share a window and its audio), which should be possible. It was also noted that for some classes of applications, we may want a way for users to allow some of these permissions at install time (for example, a remote desktop application asking permission on every start can be annoying). This is already possible with Flatpak manifests (which are static, but we might need to add some more options here), and there is a potential entitlement system being discussed (for server-driven overrides to be distributed for malicious applications, for example). Encapsulation and Collections One topic that came up last year is the ability to encapsulate a group of nodes such that they appear as a single node to other applications in the system. This could be useful for: Collapsing all the output from an application so it appears to be providing a single stream Grouping all the filters for a sink or source node, and making it appear as a single node with all the processing hidden away One piece to making such a system possible is to have a first-class notion of this group. Julian has an implementation of such an entity, called a "collection". This is currently implemented on top of PipeWire metadata, but we agree that this is likely worth having an explicit PipeWire interface for. Once that is in place, we discussed the possibility of having a smarter "proxy" node that can act as the interface that translates from the "outside" of the encapsulated region to the "inside", so that format selection, volume changes, etc. can properly be proxied to the underlying device, for example. Tooling improvements It was noted that the tools we have (such as pw-top and pw-dot) can make it hard to get at some information, such as negotiated formats, rates, etc. They can also be "noisy" when we have a large number of filters and loopbacks. While we did not have a concrete plan to tackle this, some of us have been playing with LLM-based tooling to generate some helper code for this sort of thing. At least my attempts have been too sloppy to share as yet, but it should be possible to get something useful with a structured approach. That's it for now. Watch this space for part 2!

- Matthias Klumpp: Introducing pkgcli: A nicer command-line interface for PackageKit (2026/06/14 06:22)
  For almost two decades, the PackageKit package management abstraction layer has shipped with pkcon as its command-line client. pkcon does its job, but it was always kind of a "testing" front-end for the PackageKit daemon rather than a tool designed for everyday use. The focus has instead been on the GUI tools, automatic system updates, GUI application managers and other front-ends. Its command names mirror the D-Bus API almost one-to-one (get-details, get-updates, get-depends), output is very plain, and there is no machine-readable mode for scripting. Most importantly though, there has been no development on it at all for almost a decade, so pkcon was stuck in its rudimentary state from that era. Since a lot of changes will be coming to PackageKit, and testing the daemon and working with it from the command-line was not very pleasant anymore in 2025/2026, I decided to modernize the tool as part of my work as fellow for the Sovereign Tech Agency last year. pkgcli is the new command-line client for PackageKit. It is built from the ground up to be pleasant to use interactively and easy to drive from scripts. Why a new tool? Of course, instead of introducing a new tool, I could have just expanded pkcon instead. The problem with that approach is that the pkcon utility has been around for so long and its command-line API had ossified so much, that rather than changing it and potentially breaking a lot of scripts relying on its quirks, I decided to introduce a new tool instead. pkcon can still be optionally compiled for people who need it in their scripts and workflows. The goals for pkgcli, and the features it now has are: Human-friendly command names. Verbs that read the way you'd describe the task, instead of mirroring the D-Bus API 1:1: show, search, list-updates, what-provides, instead of get-details and friends. Readable, colored

output by default (still respecting NO_COLOR and degrading gracefully). A real scripting mode. A global --json flag emits JSONL instead of fully human-readable output when possible, to make it easier to use the tool for scripting purposes. Sensible defaults. A few defaults have been changed, such as the metadata cache-age, or automatic cleanup of unused dependencies being enabled by default. This is more in line with current defaults by other tools and frontends. We also print package information in a slightly different, more readable way. Better handling of internationalized text. Text should now align properly in the terminal window, and we should no longer have completely chaotic text output on non-English locales (especially Chinese/Japanese). Why not pkgctl? Originally, this tool was called pkgctl, to match other common cross-distro tool names. However, that name was already taken by an Arch-specific distro development tool. When this issue was raised, we decided to just rename our tool to pkgcli with the next release, to avoid the name clash on Arch Linux. Examples! Here are some examples on how to use the new tool (some of which include the abridged output pkgcli prints). Search for anything containing the string “editor” in name or description, then look at the details of one result: $ pkgcli search editor Querying [] 100% ▣ ace-of-penguins 1.5~rc2-7.amd64 [debian-testing-main] ▣ acorn-fdisk 3.0.6-14.amd64 [debian-testing-main] ▣ ardour 1:9.2.0+ds-1.amd64 [debian-testing-main] ✔ audacity 3.7.7+dfsg-1.amd64 [manual:debian-testing-main] ✔ audacity-data 3.7.7+dfsg-1.all [auto:debian-testing-main] ▣ augeas-tools 1.14.1-1.1.amd64 [debian-testing-main] ▣ emacs 1:30.2+1-3.all [debian-testing-main] ▣ gedit 48.1-9+b1.amd64 [debian-testing-main] ▣ gedit-common 48.1-9.all [debian-testing-main] ▣ gedit-dev 48.1-9+b1.amd64 [debian-testing-main] [...] $ pkgcli show nano Package: nano Version: 9.0-1 Summary: small, friendly text editor inspired by Pico Description: GNU nano is an easy-to-use text editor originally designed as a replacement for Pico, the ncurses-based editor from the non-free mailer package Pine. [...] URL: https://www.nano-editor.org/ Group: publishing Installed Size: 2.9 MB Download Size: 646.0 KB Search only within package names rather than descriptions: $ pkgcli search name python3 Check for updates. refresh updates the metadata, then list-updates reports what’s available: $ pkgcli refresh && pkgcli list-updates Loading cache [] 100% ▲ cme 1.048-1.all [debian-testing-main] ▲ gir1.2-gdm-1.0 50.1-2.amd64 [debian-testing-main] ▲ imagemagick 8:7.1.2.24+dfsg1-1.amd64 [debian-testing-main] ▲ imagemagick-7-common 8:7.1.2.24+dfsg1-1.all [debian-testing-main] ▲ imagemagick-7.q16 8:7.1.2.24+dfsg1-1.amd64 [debian-testing-main] ▲ libdlrestrictions1 0.22.0.amd64 [debian-testing-main] ▲ libfftw3-bin 3.3.11-1.amd64 [debian-testing-main] ▲ libfftw3-dev 3.3.11-1.amd64 [debian-testing-main] Explore relationships between packages: $ pkgcli list-depends inkscape # list what inkscape depends on $ pkgcli list-requiring libappstream5 # list what requires libappstream5 Find the package that provides a capability, here the AV1 GStreamer decoder: $ pkgcli what-provides "gstreamer1(decoder-video/x-av1)" ✔ gstreamer1.0-plugins-bad 1.28.3-1.amd64 [auto:debian-testing-main] You can also have JSON output for most commands! Attach --json to any query and pipe the result straight into jq. Each line is a self-contained JSON object: $ pkgcli --json list-updates | jq -r '.name' cme gir1.2-gdm-1.0 imagemagick imagemagick-7-common imagemagick-7.q16 libdlrestrictions1 libfftw3-bin libfftw3-dev libfftw3-double3 Try it pkgcli is built by default alongside the rest of PackageKit since PackageKit 1.3.4. If your distribution ships a recent enough PackageKit, it should already be on your PATH. You can read its man page man pkgcli for more information. Feedback, bug reports, and patches are very welcome.

- Christian Hergert: Testing Keyboard Input Latency (2026/06/13 09:47)
  I occasionally see people go through great effort to do end-to-end testing of keyboard input latency. That is fantastic but it requires hardware and patience I don’t, nor will ever, have. Here is a much simpler way to get about 90% of the value. For example, everything but driver/interrupt handler latency and display link scanout/monitor visibility latency and of course your app side (but you could theoretically rig this up to do that

too, inside your app). Not that those aren't important, but they definitely fall into the category of things I personally cannot control for you. Keyspeed is a very simple GTK application which uses /dev/uinput to synthesize keypresses. Since it knows the time of provenance, it can compare that to when it gets the event back from compositor delivery. Wrap all that data up in Sysprof capture marks, pull in some from the compositor (GNOME Shell/Mutter support this), tie in some callgraphs/flamegraphs, and you have a very good overview of what is going on during your keypress. Run it like this (but remember to chmod back when you're done less you have attack surface available). $ sudo chmod 660 /dev/uinput $ git clone https://gitlab.gnome.org/chergert/keypress $ sudo dnf install sysprof-devel libinput-devel gtk4-devel $ make $ sysprof-cli --gtk --gnome-shell capture.syscap -- ./keyspeed $ sysprof capture.syscap Currently, this only shows you keypress send to receive in GTK, but if someone cared enough, you could make it take the next GtkFrameTimings and use that to get the presentation time. I don't need that for what I'm doing, so it doesn't. If you go to the marks section, you can dive in to a specific keypress/release cycle. Zoom in on just that section, switch back to callgraph/flamegraph profiler and see what was going on. Pretty simple, no special hardware needed. You can see how long it took, where time was spent, and more importantly, how much time was empty between things that matter.

- This Week in GNOME: #253 Fellowships (2026/06/12 20:33)
  Update on what happened across the GNOME project in the week from June 5 to June 12. GNOME Foundation marimaj reports The GNOME Foundation has selected the first recipients who will receive funding through its new Fellowship program, and is delighted to announce that Peter Eisenmann and Sophie Herold will begin work as our first Fellows in July. Sophie and Peter are both long-running GNOME contributors, with many significant contributions as members of the GNOME community. Sophie is known as developer of apps, libraries, and websites, including Loupe, Pika Backup, Glycin, and welcome.gnome.org. Peter is a long-standing Nautilus maintainer (officially known as the Files app), as well as an experienced contributor to platform libraries, including GTK and GLib. Full announcement: https://blogs.gnome.org/foundation/2026/06/11/announcing-our-first-fellows/ Miscellaneous Philipp Sauberzweig reports I'm excited to share with you that I'll be joining the Sovereign Tech Agency as a fellow for the GNOME Design Team starting in July. Check out the announcement and the full cohort of fellows in the official blogpost. During my two-year fellowship, I will support GNOME maintainers and developers with design feedback and reviews, create mockups, and coordinate efforts to standardize design patterns. My focus will be on increasing the visibility of design work, improving documentation, tools, and templates, and supporting the onboarding of new contributors. I'm looking forward to meeting many of you (again) at GUADEC and I'll post more about my activities soon. I'm currently wrapping up projects from my previous job and taking some time off. See you all back in July. GNOME Core Apps and Libraries Maps ↗ Maps gives you quick access to maps all across the world. mlundblad reports Maps now shows departures (and arrivals) for public transit stops/stations (when data is available in Transitous) GNOME Circle Apps and Libraries Tuba ↗ Browse the Fediverse. GeopJr □⚧□□ says This week, Tuba was ported to Android, opening up exciting new opportunities, as we get closer to the next release! Third Party Projects Jiri Eischmann reports Meshy 26.06 has been released. It's the first stable release of MeshCore client for Linux written in Python, GTK 4, and libadwaita. It has a feature parity with the official client, but provides native look & feel striving for the best possible integration with the Linux desktop, primarily GNOME. You can install stable releases of Meshy from Flathub or developoment releases from the app's flatpak repo. Stable releases are planned at a rate of once per month. Anton Isaiev announces RustConn 0.16 Released RustConn just turned one year old, and 0.16 is out. Thank you to everyone who uses it and files reports - that feedback shapes every release. The biggest change this cycle is the move to IronRDP 0.15: bulk compression for lower bandwidth on slow links, slow-path rendering (XRDP and older Windows no longer show a blank screen), and better compatibility with GNOME Remote Desktop. Huge thanks to the

IronRDP project and to all the open projects RustConn builds on for its connections. Other highlights, all requested by users: Connections with notes now show a small badge in the sidebar, so you can see at a glance which entries have documentation (search also takes into account the content of notes). Snap packaging caught up with the Flatpak build. A lot of macOS fixes (Keychain, SSH password auth, tray). A new Windows / WSL2 setup guide. Plus many small GNOME HIG refinements across the settings dialog and sidebar. Homepage: https://github.com/totoshko88/RustConn Flathub: https://flathub.org/apps/io.github.totoshko88.RustConn austin says Gelly 1.6 was released this week. Gelly is a Jellyfin and Subsonic/Navidrome compatible player using GNOME technologies. The last month has seen an uptick in development and contributions, with a few major features added: Gapless playback UI polish, including a new compact mode NFC card support with the companion gelly-nfc project Favorites Internationalization Gelly needs help with translations! Please see the README for details. Thanks to all that have already contributed! https://flathub.org/en/apps/io.m51.Gelly Ans Ibrahim announces Memento, the movie and tv tracking app, got updates this week with version 1.3.0: Backup and Restore functionality was implemented Import from Ticketbooth is supported Import for Movary plays and watchlist is supported Users can add plays without date Fractal ↗ Matrix messaging app for GNOME written in Rust. Kévin Commaille reports Fractal 14 has landed and is packed with lots of small changes, that make for an even better experience. Here is a quick reminder of the changes since Fractal 13: Call rooms are identified with a camera icon in the sidebar and show a banner to warn that other users might not read messages in these rooms. Calls are rendered in the timeline and incoming calls trigger a notification. We still don’t support calls, but at least now you know when someone is calling and can open another client to answer. While we still support signing in via SSO, we have dropped support for identity providers, to simplify our code and a have a closer experience to signing in with OAuth 2.0. The sidebar room filter has been improved: Enter goes to first room result, and there’s an empty state when no results match the term. The performance of the room list has also been improved, it should be mostly noticeable for accounts that have joined a lot of rooms. Informative events (Unable to decrypt, server notices…) are now styled differently to reflect their special nature and differentiate them from regular text messages that anyone can send. Sending files & location is properly disabled while editing/replying, as it doesn’t work anyway. As usual, this release includes other improvements and fixes thanks to all our contributors, and our upstream projects. We want to address special thanks to the translators who worked on this version. We know this is a huge undertaking and have a deep appreciation for what you’ve done. If you want to help with this effort, head over to Damned Lies. This version is available right now on Flathub. This cycle, we were lucky enough to get a higher than usual number of new contributors. Most of the changes listed above come from them. If you want to join the gang, you can start by fixing one of our newcomers issues. We are always looking for new members! Shell Extensions Carlos Jiménez reports New version of Gnome Football extension, now with a calendar panel integration: https://github.com/carlosjdelgado/GnomeFootball/releases/tag/v2.0.0 Arnis (kem-a) announces Kiwi (is not Apple) brings a bit of macOS feel to GNOME without getting in the way of the desktop you already use. The idea is to help people coming from macOS settle into Linux and feel at home on GNOME right away. It is fairly feature-rich and modular: macOS-style window control buttons, window controls and titles in the top panel for maximized apps, battery percentage when you are getting low, and a calendar moved to the right with notifications tucked into Quick Settings, accent colored menu entries, among others. The restyling is deliberately minimal, so it plays nice with default Adwaita theme and the rest of your setup. The latest update v1.7.0 improves the dynamic blur implementation (of course, there is blur) to the top panel, dash via Dash to Dock, and the Overview background, and adds a smoother, slowed workspace switch transition. Install it from GNOME Extensions or get it from GitHub Just Perfection says A new rule has been added to the EGO review guidelines to prevent GNOME Shell extensions from including unnecessary keys in metadata.json. 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!

- Laureen Caliman: Extending Libipuz (2026/06/11 20:49)
  From white-boarding my ideas on a Google Doc, to writing a formal design document in Crosswords, my ability to communicate technical ideas clearly is being put to the test. Writing documentation is critical to guide others' understanding of the code and choices made on a particular codebase. Especially when several developers are introduced to the system, a way to reference material leads to more preparedness to contribute to the codebase. I wrote a design document introducing the concepts I would like to implement towards creating a way to generate a dynamic grid. Critique is welcome. Standard libipuz crosswords currently rely on using an existing dictionary to fill a static box of X length x Y width. However, the implementation of vocab puzzles goes against this logic and instead generates a new grid of N length x M width based on a list of 0 <= W <= 30 words of 1 <= L <= 25 characters long. I reconsidered the idea of using a GArray to store unplaced words because I want something idempotent. To avoid unwanted time complexity bloat, the backend should not carry the memory of unplaced words. Instead, the frontend will compare the generated grid against the original list to manage words that couldn't be placed. Integrating this new feature will be a fascinating technical challenge. I created a new IpuzVocab class which inherits from IpuzCrossword. I learned how GNOME manages its developer documentation by writing a file myself to introduce this class. Writing this document made me think about the whole picture: how vocab puzzles handle grids, clues, and guesses, comparing it to standard crossword puzzles. I wrote the support to display a vocab puzzle in light and dark mode, with my next goal to integrate them via gi-docgen.
- GNOME Foundation News: Announcing Our First Fellows (2026/06/11 08:49)
  The GNOME Foundation has selected the first recipients who will receive funding through its new Fellowship program, and is delighted to announce that Peter Eisenmann and Sophie Herold will begin work as our first Fellows in July. Sophie and Peter are both long-running GNOME contributors, with many significant contributions as members of the GNOME community. Sophie is known as developer of apps, libraries, and websites, including Loupe, Pika Backup, Glycin, and welcome.gnome.org. Peter is a long-standing Nautilus maintainer (officially known as the Files app), as well as an experienced contributor to platform libraries, including GTK and GLib. Both Fellows will spend time working to enhance the long-term sustainability and health of the GNOME project. Sophie will be working to establish a new RFC process for GNOME, which will enhance our project-level governance. She will also be working on more maintainable and secure libraries through Rust adoption. Peter will work to modernize many aspects of the Files app, including thumbnailing, user directory localization, and the use of modern GNOME platform conventions. Congratulations to Peter and Sophie – we're genuinely excited to see what you'll achieve as our first Fellows, and proud to be supporting your work. We'd also like to take this opportunity to thank everyone who submitted applications to the first round of the Fellowship. We received some genuinely excellent proposals, and would strongly encourage unsuccessful applicants to apply again in future rounds. Peter and Sophie's work is made possible by the generosity of GNOME's supporters. If you'd like to help fund future rounds and support contributors like them, please consider donating.
- Jakub Steiner: Welcome to the Icon Designer Webring! (2026/06/10 00:00)
  Terry Godier wrote a beautiful essay "The Boring Internet". The internet isn't dying, he argues, just the commercial veneer glued on top of it is. Underneath all the engagement metrics and algorithmic feeds, there's still an older, slower, more federated web. One built on protocols nobody owns. RSS feeds still work (thank you, Aaron), people can set up websites and blogs. Lets start a webring in 2026 Don't worry, I haven't pushed too many pixels and gone a little cuckoo. But it's a fun exercise to remind what the web once was. We'll silently skip over the fact that I actually

started using gopher first, but even web surfing didn't begin on a search engine back in the day. It was web rings, later followed by index sites. Start Not long ago I posted about designing app icons for 3rd party GNOME app developers. The post generated quite some buzz and some old and new faces started showing up to help with the backlog. So obviously I'd like to take you on a webring tour of all the designers responsible for making the GNOME app ecosystem a little less awkward to browse on Flathub. Let me introduce you to Brage. He's been around for a couple of years now, helping to tame the flames of the reddit community, helping with the GNOME Circle project to improve the quality of GNOME apps in the wild, creating illustrations for initial states in apps, authoring some noteworthy apps himself. So thank you, Brage, welcome to the 90s! Next Up: Brage Fuglseth

- Sriram Ramkrishna: Linux App Summit 2026 Social Media Retrospective (2026/06/09 23:18)
  Linux App Summit 2026 Social Media Retrospective This is my personal retrospective post – there will likely be some version of this that will go out to various stakeholders. I want to start off by giving huge praise to our organizing team that worked really hard this year in putting this event together. Couldn’t ask for a better team to work with. Our organizing team is a mix of KDE and GNOME people. This post will focus on the outreach, fundraising, and social media campaign since that was the bulk of the work I did for LAS this year. Linux App Summit (LAS) for those who don’t know is a conference organized around the goal of encouraging developing apps on the Linux platform. With the advent of technologies like Flatpak, we had the technology to be able to ship apps directly to users instead of through the distros. Opening an opportunity for a bi-directional relationship between app developers and the users of their apps. This year marks the 10th year I’ve been involved in organizing LAS and its previous incarnation, LAS GNOME. LAS is organized jointly by GNOME and KDE who help fund and promote the conference jointly. It is a showcase of how we can unite and do impactful things. I was not able to attend this year due to other commitments. I hope other who did attend will weigh in and let us know how it was in-person. Let’s get to it! Challenging Myself I wanted to challenge myself this year and really bring in the kind of engagement that I could be proud of. I’ve not really had the kind of time I wanted to work on this and it was time to really focus and see what could be done with a proper plan. The goal I wanted to take myself is driving awareness and growing attendance on what our app ecosystem is doing. What does that entail? Improving our social media game The underlying problem I have identified is that Linux and apps was not getting into the headspace of developers. It still felt that this conference was unknown even in our own spaces. We need to break out of Mastodon and start exploring different platforms and content. In previous years, we were using Buffer since it was free but it was really difficult and unwieldy. We could only schedule 3 days in advance and at times the posts would just drop. We needed to first change the tools we used to really improve our engagement with the world. With help from the sysadmins at the GNOME Foundation (thank you, GF and Andrea Veri and Bart!), we were able to install Mixpost a self-host social media platform. The two great things about Mixpost is 1) analytics of the posts and social media platforms we were active on and not so active on 2) have a workspace around all our social media accounts and have a team of people working in it with an editorial flow and content calendar. This allowed us to share the workload of posting among many members. For instance, when I wasn’t around, Aryan Kaushik was able to take it over and post. Mixpost is now also being used for various GNOME’s accounts as well. The software continues to improve and hopefully they’ll get around to single-signon support. With the ability to actually have metrics, the next step is to actually take goals for each of the social media accounts we had and see if we could meet them. Below is a table of the targets I took and the results from February 2026 – May 2026. Instagram was actually started in the beginning of May. Social Media Start Count Target Count Result Mastodon 596 796 926 LinkedIn 534 700 619 BlueSky 0 100 39 YouTube 1420 N/A 1610 Instagram 0 N/A 42 Overall, I think we did ok! The high count for Mastodon was because of the great work of the GNOME and KDE accounts on mastodon boosting our posts

and helping promoting them before, during, and after LAS. I noticed doing things like polls on mastodon got a lot of attention without needing boosts from the other accounts. We had decent engagement on LinkedIn. Certainly better than in the past. The trick though is that LinkedIn requires a different lens when you post. Since it is mostly focus on B2B and B2C type of messaging you need to write them differently. I didn't do it this time because writing social media posts is hard and takes a lot of time and thought. I didn't take any goals for YouTube since we did not conceive that we would create content targeted for YouTube. In a spur of the moment, I did a 'podcast style' conversation between Matthias Clasen and myself talking about LAS. That gave us about 354 views. Which was encouraging and gives us some idea how organic content on YouTube would be received. Bluesky was a new account for us. So we started with zero. We gained 39 followers. That might not seem like a lot given the time frame but BlueSky is an interesting platform when it comes to engagement. You can get quite a bit of engagement even if the follower count is low. I think given more time on the platform we'll be able to make that 100+ if we keep posting content. I think hashtags matter here and playing with the right kind of hashtag and content matters. Bluesky is also was a great experiment when you didn't have big accounts like GNOME and KDE boosting you. The media partners we had 9to5Linux, Tuxdigital, It's FOSS, and Linux Magazine all helped in this regard by using their accounts boost our posts in these other platforms and give us visibility. Thank you to our media partners for helping out and we hope we can work with them closer next year. I'll like to engage with them further to see how we can help each other out including contributing content. Another idea is to reach out to the speakers of these talks and get them to write some articles that could be contributed based on their talks. Finally, Instagram. This is an untapped gold mine. I was skimming through the platform looking for GNOME/KDE/Linux desktop type posts to see how well content did. Saw one young lady, who showed off her GNOME desktop with some caption and it gave her 130k views. It was about 10 seconds long. That was impressive. I posted a short video talking about Linux App Summit, and while I got about 130 views – the analytics said most stopped watching after 9 seconds of the 4 minute video I posted. That hurt my pride. I resolved to do better and get better engagement with a 10-15 second video that packed more information and visually more stimulating. As of now the LAS account is still gaining followers despite not posting for 2 weeks. Once again, the media partners helped by liking my posts while the other accounts lay idle. Working with YouTube Influencers One other aspects of my plan on boosting the visibility of LAS was to start working with influencers on various platforms. I made a few attempts with a few I knew but was only able to get on one podcast – Tux Digital. Michael Tunnell was kind enough to invite Aleix Pol and myself on his show. For an hour and half, we answered questions and did some bantering. We even went in some organic directions that was fun! I know I had fun, I hope Aleix did too. The exposure was pretty good with approximately 8k+ views for that episode that was 90 minutes. The feedback to the video was very positive with many resolving to attend the conference. Unfortunately, I didn't set up utm links so that I know where people came from. Through social media and influencers, we hoped to break out of our media ecosystem and branch out to platforms that developers and Linux enthusiasts hang out and consume content. Meeting where developers are needs to be something we will need to focus on going forward. Results The in-person conference was a success, we had 110 people at the conference, the venue capacity was 100. We had 156 people who registered for the conference, this is about a 71% conversion rate. The industry average for free in-person events is 50%. For LAS, this is unprecedented because we usually had a much worse turnover rate historically. At one point, a few years ago I had started looking into doing registration fees to give people some reason to go and not ghost the conference. For online this year, we had about 50 online registrations but it's hard to gauge anything about online participation since we freely published the YouTube link on social media. The results for the conference for online had the following results on YouTube: 2025 2026 Day 1 Views 922 1.7k Day 2 Views 485 1.5k The above numbers show views within the 24 hour period of each day. These are really good numbers where we've more than doubled our

viewership on one of the two days compared to last year. Ostensibly, it shows that our social media did build awareness. Here is the (still increasing) numbers as of now on Youtube: First Day: 2k views Second day: 2.7k views The videos are currently being broken up into individual talks. But the individual talks as of now are averaging about 300 views or so with the top one being 900+ views for Lennart's keynote. The aggregate views for all the 13 individual talk videos posted so far is 4.9k. If we combined that with the combined 2k and 2.7k views, we can simplistically (mathematically speaking) would be 9.6k. Interestingly enough, that is not a big difference from the 8k views of Tux Digital podcast that Aleix and I were on. □ In regards, to the people who attended Linux App Summit, 16 people filled out the surveys and by in-large most of the people who attended heard about LAS through word of mouth and not so much through social media. So, that's an interesting data point. I expect that is because of people like Lorenz Wildberg (thank you!) who did on the ground outreach. Interestingly, enough our online views were quite good compared to say SCALE which peaked at 7.1k views for one talk but in general the average views was less than the average views than at LAS. Our subject matter is increasingly important. Also to be clear, views do not translate to people. Looking towards 2027, we'll want to increase our in-person attendance while doubling our online views. Something we will be focusing on when we organize for next year. Next Steps Our organizing team will be posting relevant individual talks on social media once all the individual videos have been posted. I hope you all share those with everyone. Secondly, we would love to add more people to our organizing team. Specifically, in order to really build out our outreach we need a lot more people to help network and reach out to developers from different communities and different platforms. This way we can start building relationships with other desktop projects, app developers, game developers, designers not just from Linux but from other platforms as well. For that we need a small army of advocates. Thanks I wanted to add the social media work is a team effort and that I would like to thank Aryan Kaushik and Aniqa Khokar for helping write posts and editing them. As well, I would like to thank Caroline Henriksen for doing all the images and themes on social media!

- Michael Catanzaro: Please Do Not Ban AI-Assisted Issue Reports (2026/06/08 21:30)
  Many GNOME projects have adopted a policy banning all contributions generated by LLMs. This policy was originally developed by Sophie for Loupe, but is now used in many other notable places: This project does not allow contributions generated by large languages models (LLMs) and chatbots. This ban includes, but is not limited to, tools like ChatGPT, Claude, Copilot, DeepSeek, and Devin AI. We are taking these steps as precaution due to the potential negative influence of AI generated content on quality, as well as likely copyright violations. This ban of AI generated content applies to all parts of the projects, including, but not limited to, code, documentation, issues, and artworks. An exception applies for purely translating texts for issues and comments to English. AI tools can be used to answer questions and find information. However, we encourage contributors to avoid them in favor of using existing documentation and our chats and forums. Since AI generated information is frequently misleading or false, we cannot supply support on anything referencing AI output. I won't attempt to argue that you should allow use of AI for writing code. If you wish to ban LLM-generated code, fine. That's probably inadvisable, but I am not going to object. But this policy is far stricter than that. Notably, it strictly prohibits AI-generated content in issue reports (except to translate text). Don't do this! Prohibiting bug reports is stupid and just makes your software worse. Please make sure your project's AI policy allows for at least AI-generated static analysis results and AI-generated vulnerability reports. Otherwise, you prohibit entirely unobjectionable problem reports. It's hard to imagine what could possibly be the value of prohibiting valid bug reports. AI-generated static analysis works well: the AI is able to think about your code, follow execution paths, and automatically discard most false positives to avoid bothering you with them, and the quality of reports is generally pretty high. They are far from perfect, but the same is true of humans. Here is a typical example of an AI-generated static analysis finding: 2. Resource

leak in update_credentials_cb on gnutls_credentials_set failure File: tls/gnutls/gtlsconnection-gnutls.c:169-172 When gnutls_credentials_set() fails, the function returns without calling g_gnutls_certificate_credentials_unref(credentials). The credentials was either freshly allocated or ref-bumped, so it leaks. Pasting this into an issue report clearly violates the ban on AI-generated content. And yet, why would you not want to receive a clear and concrete bug report for memory leak? I understand not all maintainers are fond of AI, but is your dislike really so extreme that you would choose to ignore valid problems and intentionally make your software worse? If not, then your AI policy should thoughtfully consider how to handle AI-generated content in issue reports. Certainly do not adopt a policy that outright bans all AI-generated content in issue reports. As an issue reporter, you could theoretically take the problem found by the AI and rephrase all the words, then claim that it is no longer AI-generated content because it is rewritten. This is a waste of time and usually results in a lower-quality, less-detailed result, but you could plausibly do that. Or, if you want to go above and beyond, you could just jump ahead to creating a merge request. But realistically, if your project does not allow any use of AI in issue reports, it's more likely that either (a) you won't receive the issue report in the first place, or (b) you won't receive such issue reports from experienced developers who read and respect your policy, while users who do not read your policy will continue to submit them. What about security vulnerability reports? Since the start of this year, I have reviewed well over 100 vulnerability reports that I strongly suspect were generated by AI. To reach the "over 100" claim, I sadly only considered vulnerability reports submitted during a particularly heavy four week period, so this is an extremely loose lower bound. Suffice to say, I have seen a lot of them. The quality varies dramatically. Vulnerability reports are now often better or worse than before: better because an experienced human working with a good AI is able to find vulnerabilities that would have surely gone unnoticed without AI, and worse because an inexperienced human with a bad AI might create some pretty terrible issue reports, a significant proportion of which are just outright spam. Low-quality reports remain a problem, but nowadays most AI-generated issue reports are quite good. Maintainers do not need to tolerate spammy vulnerability reports. If an issue report is bad, of course go ahead and close it. If it's really bad, then I sometimes don't even bother replying. But banning good vulnerability reports solely because some portion of the report was generated by AI is unacceptable. AI-assisted vulnerability reports are the new industry standard, and this is not likely to change. Prohibiting issue reports reduces the quality and safety of your software, punishing your users. This is too extreme.

- Jussi Pakkanen: Faking keyword arguments to functions in C++ (2026/06/08 12:09)
  One of the many nice language features in Python are keyword arguments. They make some types of APIs concise and readable. Like so:Unfortunately C does not have keyword arguments and, by extension, neither does C++. Adding them as a language feature would take 15-20 years of effort, most of which would consist of trying to convince people via email that such a feature is important and should be added.There have been attempts to implement this via macros and template magic (link), but they have not seen widespread usage probably because they are using macros and template magic. However it turns out that with modern language features you can fake keyword arguments fairly convincingly. Like so:The add_argument method takes a single argument which is a struct. The extra curly braces inside the parentheses boil down to "whatever the underlying argument is, construct it in place with these parameters". The dotted names are designated initializers, so those fields get the specified value whereas other fields get their default values.And there you go, keyword arguments in C++. You just have to squint a bit and pretend not to see the extra curly braces.
- Ivan Molodetskikh: Using Fedora Silverblue for Compositor Development (2026/06/05 12:37)
  I've been using Fedora Silverblue on my desktop and laptop for the past, what, five years? Silverblue is Fedora's main atomic variant, a spiritual counterpart to Fedora Workstation. I also make niri, a scrollable-tiling Wayland compositor. In other words, a core system component that you

cannot properly test from inside a container or VM—you really want it directly on the host. So, why would I choose an… immutable distro? How does that even work? Fedora Silverblue makes a frequent occurrence in my niri release notes screenshots. Atomic distributions have been slowly rising in popularity. Their main selling point is reliability: upgrades work by swapping the old system for the new one in one go across a reboot, rather than modifying the files in-place. Package conflicts and other errors are caught at the time of assembling the new version (in a separate folder), and therefore cannot break your running system. And if a successful update turns out buggy, atomic distros let you simply reboot back into the old version and keep using it as if nothing happened. This “being able to reboot back” thing becomes even cooler once you realize that it works even across major distro upgrades! When the next Fedora Beta rolls around, I can just rebase my system on top of it to kick the tires, and if anything is broken, I can simply reboot back to stable Fedora (and then undo the rebase). This is like learning about source code version control. A big weight off your mind any time you want to mess around with your OS. You can just go back. So, by now there are plenty of atomic distributions to choose from. There’s a whole host of Fedora atomic desktops, Endless OS, the gaming-focused Bazzite and other Universal Blue images. GNOME OS Nightly is atomic, as well as SteamOS powering the Steam Deck. Many of these are built with OSTree which is something of a “git for operating system binaries”. But, you may ask. What if I develop these operating system binaries? Aren’t atomic distros immutable and all, how do I test my work? Turns out, this is not a problem at all! In fact, the same tech that lets you go back after an update can also let you freely tinker with your host system and safely go back after a reboot. I’d say that thanks to this ability, atomic distributions provide even more benefit for system component developers than for others, given that they’re constantly testing changes that may break their install. So, let me show you how I do compositor development on Fedora Silverblue. We’ll start with toolbox where most of the work happens, then proceed to the fun stuff. Toolbox # On your immutable host system, you need a place where you can install the development environment. Fedora Silverblue comes pre-installed with Toolbox, which provides just that—a terminal in a normal, mutable Fedora where you can sudo dnf install to your heart’s content. Under the hood, it’s just a podman container with a whole range of things auto-mounted from the host: the Wayland socket, networking, devices, D-Bus, and everything else needed for apps to “just work” as much as possible from inside the container. You can even interact with it through podman commands: ┌ ~ └─ podman ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 6ceccce5581e registry.fedoraproject.org/fedora-toolbox:44 toolbox --log-lev... 2 months ago Up 41 minutes fedora-toolbox-44 Most of your development work happens here. Install all the libraries, compilers, editors, LSPs, debuggers, and the rest of the kitchen sink. Since all of this resides inside the same container, it can all talk to each other and work together. One slightly annoying detail is that since your fully-configured editor is inside the toolbox, you can’t use it to edit files accessible only on the host (e.g. configs in /etc—the system inside the toolbox has its own files there), but that is honestly a fairly minor problem in practice. Fedora Silverblue comes with nano, which works, and if editing host-only files is a frequent occurrence for you, you can always rpm-ostree install a more featureful editor. Another annoying problem is that currently, toolbox prevents SIGHUP from reaching apps, so if you run your favorite editor then close the terminal window, it will happily keep running in the background (along with all its rust-analyzers and such, eating several gigabytes of RAM). So, running things in a toolbox works perfectly well for most development. CLI tools will run fine, GUI apps will run fine, you can build and install libraries inside the toolbox and test them on apps inside the same toolbox. Even with Wayland compositors, most of them can run as a window (gnome-shell --nested, or simply sway or niri), which is enough to test the majority of the code base. Moreover, since ~2023, toolbox exposes everything necessary to run compositors on a TTY directly. You can switch to a different VT with CtrlAltF3, toolbox enter, then start a compositor, and it will work as is. This way you can test different input devices directly (trackpad, tablet, touchscreen), test monitor and GPU handling, do proper performance profiling, and so on. Just remember to

install a terminal and some GUI apps inside the toolbox because launching the host ones into a toolbox compositor is a bit annoying. While toolbox is somewhat Fedora-specific, for everything else there's distrobox. It's a separate project, but by and large has the same idea—let you easily install different distros as podman containers with automatic host integration. I mainly use it to build or test things on Arch, but I imagine most of what I wrote above works just as well with distrobox. What if this isn't enough, though? Say, you're working on a component like NetworkManager or systemd that must run on the host system. Or, you want to be able to log in to a test build of your compositor along with the rest of the full desktop session. Let's look at an easy way to do that. Unlocking the host # Run sudo ostree admin unlock, also known as rpm-ostree usroverlay.12 This will mount a mutable overlay filesystem over /usr for you to play around in. The overlay will last until the next reboot, at which point you'll be back to a clean working system. Now you can simply sudo cp your development build into /usr/bin and restart the service you're testing. This also works with libraries. Say, you want to test your changes in GTK against apps installed on the host.3 Build it inside the toolbox, then copy the binaries to the (unlocked) host, and there you have it. Binary compatibility is generally not a concern since Silverblue updates daily and very closely matches the regular Fedora that you build against inside the toolbox. sudo cp is not a proper substitute for installing though, and you cannot use it as easily for many projects. So let's get some proper tooling on the host. Layering development tooling # Contrary to an apparently widespread belief, you can install packages on the host in Silverblue. This is called layering and is a perfectly normal and supported operation, primarily useful for adding system components such as terminals, window managers, or GPU drivers. Running rpm-ostree install alacritty will cause rpm-ostree to install, or layer, this package on top of the base Silverblue image every time it updates. After a reboot, you'll have Fedora with Alacritty, as if you installed it on a regular, non-atomic system. If the change is sufficiently non-invasive, running sudo rpm-ostree apply-live lets you skip the reboot and have a newly installed program available right away.4 When should you layer (as opposed to installing in a toolbox)? Layering is more annoying and slower, and misses the benefit of throwing away a toolbox to start fresh. So, I limit layering to programs that must run on the host, and tools that I frequently need on the host. Here's my list of layered packages that's been more or less unchanged for several Fedora releases: ┌ ~ └─ rpm-ostree status State: idle Deployments: fedora:fedora/42/x86_64/silverblue Version: 42.20250824.0 (2025-08-24T02:55:42Z) BaseCommit: d58dc92e5b05b6a95a0d9352edd864f1292c1883b9b32ac2e6f0af1a2263395a GPGSignature: Valid signature by B0F4950458F69E1150C6C5EDC8AC4916105EF944 Diff: 12 upgraded RemovedBasePackages: firefox firefox-langpacks 142.0-1.fc42 LayeredPackages: alacritty distrobox dnf fastfetch fish foot fuzzel gamescope gdb gnome-console google-roboto-fonts htop hyprlock i3 kanshi labwc langpacks-ru lm_sensors lxqt-policykit mako nautilus-python netconsole-service niri perf quickshell-git rocminfo strace sway syncthing sysprof tmux trash-cli waybar wlsunset LocalPackages: edid-asus-1-1.fc34.noarch Initramfs: --include /etc/initramfs-overlay / In this output, you can find: I removed Firefox with rpm-ostree override remove—I prefer the official build from Flathub. Terminals (must run on the host to access the full host filesystem5): alacritty, foot, gnome-console. My preferred shell: fish. Tool I frequently need: tmux. Services and tools that I want to run without a toolbox: syncthing, distrobox, netconsole-service, trash-cli, htop, fastfetch, lm_sensors, rocminfo. Desktop components: fuzzel, hyprlock, i3, kanshi, labwc, lxqt-policykit, mako, quickshell-git, sway, waybar, wlsunset. edid-asus and the initramfs-overlay provide the EDID for one of my monitors after AMDGPU broke it back in kernel 4.19.6 Along with these, I layer several development tools: gdb, strace, perf, sysprof. These frequently come in handy whenever I need to debug or profile programs running on the host (or do full-system profiling in case of Sysprof). And then there's dnf. What? Layering dnf # What is dnf, a regular Fedora package manager, doing on an immutable Silverblue host system? By itself, it's not very useful indeed, since it can't modify /usr. (Though, it can dnf copr enable, which is convenient. rpm-ostree copr when?) Where dnf on the host shines, however, is when you combine it with sudo ostree admin unlock. After unlocking, you can

install whatever you need in the moment with dnf. This is much faster than rpm-ostree, never requires a reboot, and in fact a reboot makes it all clean up and go away, since it was all in a transient /usr overlayfs. Example workflows: dnf debuginfo-install to debug/profile something on the host with symbols, report crashes, etc. dnf install some host-only program to test it. Follow up with rpm-ostree install if you decide to keep it. dnf builddep gtk4, then build and sudo ninja install GTK 4 right on the host to test it against host apps. If anything breaks, just reboot, and you're back to a clean working state. Unlocking + layering dnf is a very powerful development workflow to the point where I'd almost want dnf included in Silverblue by default. Unfortunately, this workflow is also unobvious enough that the dnf maintainers accidentally prevented it from working some time ago (thankfully, quickly corrected). I understand the UX concern about having dnf visibly available when it cannot work outside this specific workflow, but perhaps Silverblue could just hide it somehow unless the host is unlocked, or rename the dnf binary? Persistent unlocking # Generally to put something persistently on the host, you'd just layer it with rpm-ostree install. However, sometimes what you want is a temporary change that also happens to persist across reboots. This sounds weird, but consider testing a kernel build. You want it to be temporary and easy to roll back, but you kinda have to reboot into the new kernel. And you also don't want to spend extra time building and layering .rpms. For this situation, ostree admin unlock comes with a --hotfix flag. It'll persist the temporary overlay across reboots, and will only reset itself once you explicitly make some change with rpm-ostree. Note that you never lose the ability to reboot into the previous, working system. Summing it all up # So, this is what my development workflow looks like. Most work happens in one kitchen-sink toolbox that I (like to but am not required to) reinstall every Fedora release to keep cruft from building up. This includes building and running niri on a TTY. After finishing a change, I unlock the host with sudo ostree admin unlock, copy over the niri binary, and re-log in to test it in my real session. This will automatically reset upon a reboot. When working on a long-running branch, I'll build a work-in-progress niri .rpm and layer it with rpm-ostree install to persist the new version across reboots. I use dnf install on the host when I want to throwaway-test something host-specific and have it automatically reset upon a reboot. Over time I made a few small quality-of-life tweaks to smooth out some rough edges in this workflow. For example, toolbox enter is a mouthful and always drops me into bash. Enter t, a script in my ~/.local/bin/, always available in $PATH: #!/bin/bash if [$# -eq 0]; then command=fish else command="$(printf "%q " "$@")" fi exec toolbox run -c fedora-toolbox-44 bash -ic "$command" Now, typing t puts me in the toolbox directly into my dear fish shell. Typing t some-program "with complex" arguments | grep "and stuff" also works as expected, with correct argument passing thanks to printf "%q ". This works for .desktop files too. Say, you installed VSCode in the toolbox and got a .desktop file. Just change: Exec=/usr/share/code/code --ozone-platform-hint=auto %F to: Exec=t /usr/share/code/code --ozone-platform-hint=auto %F and it'll run in the toolbox. (I understand distrobox handles .desktop files automatically.) Note that I use toolbox run but route the command through bash. This is necessary to get all environment variables like $DEBUGINFOD_URLS that distros keep stubbornly putting in /etc/profile.d/ scripts, which of course don't get sourced without a bash -i. Another quality-of-life improvement was binding a separate hotkey to spawning a terminal directly in the toolbox. I actually noticed that most of the time, when I open a terminal, I want to be in the toolbox, so now my SuperT spawns the toolbox Alacritty, while the less convenient SuperShiftT spawns the host Alacritty. Furthermore, at some point I got tired of waiting for the… ┌ ~ └─ hyperfine -w 3 --shell=none 'true' 't true' Benchmark 1: true Time (mean ± σ): 411.9 µs ± 35.8 µs [User: 248.9 µs, System: 111.3 µs] Range (min … max): 374.1 µs … 1147.6 µs 5794 runs Benchmark 2: t true Time (mean ± σ): 257.8 ms ± 2.0 ms [User: 3.0 ms, System: 6.1 ms] Range (min … max): 255.2 ms … 260.5 ms 11 runs Summary true ran 625.92 ± 54.60 times faster than t true …extra 250 ms for toolbox run, and wrote a script that keeps Alacritty running as a daemon inside (and outside) the toolbox, making opening a new terminal window always instant. As a bonus, this happens to fix the SIGHUP problem that I mentioned above: since Alacritty runs directly inside the

toolbox, closing its window will properly close the terminal app running inside. (Eventually I went even further and made a tiny service for fun that runs inside the toolbox, listens to a socket, and runs the command it receives. I only use it in .desktop files though instead of t to avoid the 250 ms delay.7) What about other systems? # I quite like my Silverblue setup. It very much works, and with the tools that it has, it lets me do anything that I might need. Silverblue is not without its problems however, so I've been thinking about what parts of the experience I find important, and how well other distributions currently satisfy them. 1. The ability to reboot to a previous, working system. Most new atomic/immutable distros can do this since it's the main value proposition. It's also possible on NixOS. On traditional distros I think you can get something close with btrfs snapshots, but it requires a complex setup. A/B updates tie closely into this, where rather than mutating the running system, an update is prepared in a separate folder, then atomically swapped with the previous system version (which remains available to boot into should something go awry). 2. Anti-hysteresis. The host system always stays clean, packages don't build up over time. On a normal distro, a few months is enough for you to scarcely have any idea about all the random one-off packages you installed and forgot about, especially various development tooling and build dependencies not to mention the texlive-full installation. They use up disk space and time during system updates, sometimes cause conflicts and other annoying issues. Config migrations build up, and your system gradually drifts away from a clean well-tested upstream state. Immutable distros solve this by not letting you install stuff on the host, and every updated rebuild of the host system starts from a fresh state, so there's no accumulation of junk. NixOS and Silverblue do let you add (layer) packages, so they can build up, but: they make it sufficiently annoying, making you prefer non-host environments such as toolbox for one-off packages; even with layered packages, the system is rebuilt from a fresh state every update. Technically, you could use toolbox for everything even on a normal Fedora Workstation, but this requires discipline and doesn't save you from config migrations, SELinux labeling changes, etc. 3. The ability to easily install things on the host. This is the part where many newer immutable distros fail to provide a good experience. I need to install programs on the host, whether it's because I want some host desktop components, or to test my own compositor, or whatever. Often, I want to install something on the host quickly. For distros such as Universal Blue spins and other bootc-based systems, the suggested way to include components on the host is making your own downstream spin. But this works only for long-term packages: I don't want to spend time editing and kicking off a full system build just to test some new terminal or notification daemon, not to mention the whole question of how to keep such a custom system always up to date with its base distro. Compare this with rpm-ostree install on Silverblue: one command, slow but tolerable, and the OS remains automatically updated with no extra setup. Some systems are even more limited, like GNOME OS which is based on the Freedesktop SDK. The selection of tools and libraries available in the Freedesktop SDK is (intentionally) much more limited compared to most distros, so in many cases you'll find yourself having to go and build whatever you need from source. If that happens to be something big and complex like Qt (to try a hot new Quickshell-based desktop): good luck; I hope you didn't have plans for the weekend. A common suggestion for these OSes is systemd-sysext that lets you build an image and overlay it over /usr. Florian Müllner gave a talk at the 2025 GUADEC showing a nice workflow for using sysexts for Mutter and GNOME Shell development and testing on immutable distros. It's also possible to enforce system version compatibility checks in sysexts. A system like GNOME OS could build and ship a collection of sysexts version-locked to the runtime they were built against, and automatically updated together with the rest of the system using systemd-sysupdate, resulting in an experience similar to layered packages. (In fact, GNOME OS does have that, just the selection of sysexts is fairly small.) Some software can be packaged into self-contained sysexts that work on most distros. The Flatcar sysext-bakery is one repository of such sysexts. What's wrong then? Well, the main limitation of sysexts is that they are meant for tools without dependencies. They do not do any dependency resolution or support any dependencies other than, optionally, the base OS itself. Back to my

example, while it’s possible to build and ship sysexts for Qt apps that bundle Qt itself, all of those sysexts will carry their own copies of Qt. Even worse, since they are mounted into the same filesystem tree, conflicting files (say, different-version Qt binaries) will get mounted only from one of the sysexts, whichever one happens to mount last. So sysexts aren’t a complete replacement for packages (nor are they intended to be). 4. The ability to make transient changes to the host. While I don’t immediately see why you couldn’t put a writable overlay on any regular distro like what ostree admin unlock does, I haven’t seen anyone doing it, or any simple “no thinking necessary” tools for it.1 Perhaps it’s too easy to mess up outside immutable systems? It’s worth noting that some paths like /etc aren’t usually covered by immutability and overlays, so you still need to be a bit careful. Conclusion # All in all, Silverblue appears to be a sweet spot between offering immutable/atomic guarantees with plenty of useful tooling bundled in, while also being a normal Fedora with a wide package selection available for both persistent layering and quick transient installation. I appreciate the QA and other behind-the-scenes work that goes into my ability to install Silverblue and be reasonably sure that it will work, and keep working, with all of my hardware, and that I won’t have to hunt for packages to get a working bluetooth or what have you. My Silverblue installs are the longest I’ve kept any single distro, and I have no urge to reinstall because my host system remains clean and I know exactly what it comprises. My issues with Silverblue mostly boil down to some rough edges and slowness of rpm-ostree, and some less than ideal Flatpak repository defaults. Having to do most of the work in a container is somewhat annoying at times, especially when dealing with nested containerization or VMs. But I’m not sure there’s a better way fundamentally, without trading host system robustness. For the few things that do require it, I can always unlock the host. I hope this post sheds some light on immutable system workflows and perhaps inspires you to try one. I’d also love to hear your feedback and suggestions! Did I miss something? Is there a better way of doing things? A new system that solves all problems and makes everything better? Please reach out to me on Mastodon or by email, linked at the bottom of the page! I’m told the modern alternative is systemd-sysext merge --mutable=ephemeral, which works across all distros and not just Silverblue. Haven’t tried it myself yet! ↩ ↩ I didn’t quite realize this before, but rpm-ostree usroverlay seems to literally exec ostree admin unlock: ┌ ~ └─ rpm-ostree usroverlay -h Usage: ostree admin unlock [OPTION…] Make the current deployment mutable (as a hotfix or development) (...) ┌ ~ └─ rpm-ostree usroverlay --version libostree: Version: '2025.4' Git: 99a03a7bb8caa774668222a0caace3b7e734042e (...) ↩ Which is, uhh, not a lot of apps come to think of it. Nautilus, Ptyxis, Software, System Monitor, Settings, xdg-desktop-portal-gnome dialogs—the rest come as Flatpaks on Silverblue. How to test your GTK changes against those Flatpak apps? Uhhhhhh ↩ For years, it’s been rpm-ostree ex apply-live, where ex stood for experimental. I guess I’ve been procrastinating on this blogpost long enough that it had time to graduate to non-experimental rpm-ostree apply-live. ↩ The Ptyxis terminal can work properly on the host even when installed as a Flatpak. It does this by spawning a small binary on the host (through a host-run permission) that does all command spawning and PTY communication, while the Ptyxis GUI remains inside Flatpak. This is a clever workaround, but requires a sandbox hole and very careful engineering, and arguably runs somewhat at odds with the point of Flatpak. ↩ Since writing that example, I replaced that monitor and finally got rid of the custom initramfs. This is faster because without overrides, Silverblue directly uses an initramfs built on Fedora servers, and I think it also works better with secure boot? Either way, I wanted to leave it in as an example that you can customize the initramfs on Silverblue if needed. ↩ See for yourself: ┌ ~ └─ hyperfine -w 3 --shell=none 't true' 'true' 'tb true' Benchmark 1: t true Time (mean ± σ): 259.5 ms ± 3.6 ms [User: 2.9 ms, System: 6.2 ms] Range (min … max): 255.7 ms … 266.6 ms 11 runs Benchmark 2: true Time (mean ± σ): 408.7 µs ± 34.2 µs [User: 248.6 µs, System: 107.1 µs] Range (min … max): 370.2 µs … 1152.8 µs 6665 runs Benchmark 3: tb true Time (mean ± σ): 462.8 µs ± 41.7 µs [User: 264.2 µs, System: 135.6 µs] Range (min … max): 399.2 µs … 786.4 µs 6688 runs Summary true ran 1.13 ± 0.14 times faster than tb true 635.00 ± 53.80 times faster than t true ↩

- Daniel García Moreno: Take it easy. A guide to avoid burnown during the Vulnpocalypse (2026/06/05 10:00)
  Do not let the AI to remove the fun part from software development. We shouldn't allow gen AI to write software just because it "can". First, we must ask if it "should" do it, and even then, we should ask if we want to delegate the fun part, the thinking, the writing, the learning. Remember what's important, journey before destination, we are the Code: Do not let AI to destroy the community, do not let it destroy the technological knowledge commons. tl;dr Open Source maintainers are dealing with a lot of new reports and pressure to "fix" the project due to generative AI. We need to find a way of stopping this and get back to something maintainable before all maintainers get burned out and look for a job in a farm: 100% secure software doesn't exists, so there will be always a possible CVE there. As Spaf said in 1989: The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts. Fixing bugs, adds new bugs, and if you need to fix something quick, the probability of new bugs will be higher. Do not forget about the First Law of Programming: If it works, don't touch it The amount of CVE reports is lowering the CVE credibility and quality, so if everything is a "high" security issue, we can't prioritize now and these reports are not different from random issues in github. Do not listen to The Boy Who Cried Wolf Stable software is sable because it doesn't change too much. It's something that we are willing to loose trying to reach the impossible of 100% secure software? The actual problem There's a lot of money in AI tech right now, and everyone is trying to make the best gen AI tool or just pretend that their tool is the best. In relation with the software analysis and writing, targeting the open source is the obvious strategy. It's interesting to scrap every line of code, patch, pull request, issue and discussion around software to train your model, so AI scrappers are DDoSing open source projects infrastructure. To promote their tools or themselves, Security Researches are using AI to target any project, reporting High security vulnerabilities, with the only goal of getting a CVE number to say how good they are. This second point is affecting maintainers, because now you are receiving a lot of poor quality security reports, that are generated with AI and that looks plausible and are hard to read. You need to spend a lot of time to check if there's an actual wolf there or if it's again this boy that's tricking me. This is burning the energy of maintainers, that instead of doing something productive are wasting their limited time talking with a Stocatic Parrot. Do not let the AI Bros to use classic manipulation techniques on you! A lot of open source projects are maintained by volunteers that do the work with passion and love. And even if it's the job that paid your bills, the maintainer can feel the pressure. When someone put a lot of love in something and work on it during years, it's part of his identity, so attacking the software is like attacking the person behind it. This is nothing new, and a lot of people take advantage of this emotional link to manipulate the maintainer to do something that he do not want to do. AI bros are using these techniques, do not let them to manipulate you and define your project agenda. Here's a (not complete) list of known manipulation techniques that you can detect (and disarm!) in your daily community work: Flooding the queue. Just create so many new issues that the actual maintainers can't deal with it. You feel responsible for the project and feel bad because your TO-DO list is growing. This software is not secure (doesn't do what I want), I will use this other one instead that's better. The classic, "GNOME doesn't allow me to change this specific preference, I'll use KDE from now on". This software is low quality, it doesn't follow the (my random) quality standards. Direct attack to the maintainer self-esteem. Gaslighting software development. LLM are expert at this and people that uses it just copy the tactic. When the maintainer detects something weird and just tries to blame the other person for reporting nonsense and wasting all people time, it starts to invent new arguments and ignore the previous interaction. So, take it easy, and remember the best clause in almost any software project, THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU: Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF

ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. Is the software more insecure in 2026? No. Anyone old enough could remember how insecure old software was. Do you remember windows 98? Do you remember the internet when everything was http (without that little s at the end), when people use ftp to logging into their server and modify the php code directly on production? It's true that today we have more dependency on technology, but it's also true that everything is more secure, we have more and better cryptography, we have different levels of isolation, virtual environments, containers, virtual machines... But we have the feeling that since AI can analyse all the software and look for vulnerabilities, we are doomed, because any stupid kid can hack my over engineered GNU/Linux machine! First, that's not true, you need to know about security to get something useful from any AI tool. But even if it was true, what can you do about it? We need to be practical and find a balance between risk and usefulness, so do not overestimate the risk just because everyone is talking about it right now. But even then, the security paranoia is not good for anyone. Software is inherently buggy, people write software and makes mistakes, so a possible vulnerability appears. In theory, these bugs are fixed when discovered, so it's always recommended to update to the latest version, because almost all known bugs will be fixed. But it's also known that new versions comes with new functionality and code, and that means new "unknown" bugs or different behavior. That's a headache, so that's why the stable and Long Term Support are popular distributions, because "if it works, don't touch it". Stable packages just get the fixes, not new features, but fixes are also code changes, so there's always a possibility to break something, even with a patch update. The stable software has a lot of value, do not let the AI security paranoia destroy that, and convert everything in a rolling release with the latest and greatest (and possibly broken) software. Sometimes it's better to keep using something old, with known vulnerabilities that you can mitigate, than use the latest with unknown new vulnerabilities that you can't do anything about. I will fight AI with AI Please, do not do that. What I was trying to argue during this long post is not a technical problem. The current burnout problem in open source is a social problem, you can't fix it with a new layer of probabilistic tokens. Community reaction against AI. The current industry push for the usage of AI everywhere is affecting a lot of people, and as a reaction a lot of people are directly fighting back. Using gen AI just sends the message that you do not care enough to do it yourself, and destroy the trust on the project. It doesn't worth it. Even if the AI works (that it doesn't) it doesn't worth it. Writing code is easier than reviewing, you learn and grow with every new line of code that you write, delegating the fun part and personal growth part to an AI will make you work more miserable and you will be a junior forever. It doesn't create community. Think about it, it's hard to get someone involved in a software project, but who will want to read or improve the code produced by a gen AI? The only future collaborator will be another AI. Take it easy Just remember, you can always say no, there's no hurry, and there's no need to work on something that you don't want just because other people consider that important. Free Source is something done by people, for people. The software is important, but the community around it is sometimes more important. We use Free source not because it's technically better (that it is), but because we trust who, how and why are writing it. Remember why are you doing this, do not remove the Fun part, continue with the Just for Fun mood.

- Nick Richards: Sunset Appearance (2026/06/03 21:50)
  I love adaptive interfaces and technology that blends in more than the average human. I’ve spent literally years tinkering with ‘frecency’ ordered lists, bought a meural screen and have recently been glorying in the fantastic GNOME Adaptive Brightness. On that last point, whilst GNOME already has Automatic Screen Brightness, and it is a good feature, dmy3k’s extension goes further on the specific machines with cool hardware:

steadier behaviour with changing light, smoother transitions and brightness curves you can tune. One of the things I've been exploring with extensions recently is 'this feature, only more so' and adaptive brightness is a good example. Living far from the equator, evenings happen. The room goes grey, the window stops being a useful light source and GNOME is still cheerfully in light mode because I told it to be bright at the time one takes screenshots. Night Light is already doing its bit by then. The display has warmed up, which is nice, but the rest of the interface lacks the level of 'darque' required. I wanted the normal GNOME appearance preference to follow the day as well: light while it still feels like day, dark once the evening has properly arrived. Users of other operating systems may be aware of this feature, but for the purposes of this blog post let us pretend that everything below is entirely unique. So I hacked up Sunset Appearance, a small GNOME Shell extension for GNOME Shell 45 to 50. At civil dusk it writes the same setting GNOME Settings uses: org.gnome.desktop.interface color-scheme = 'prefer-dark' At civil dawn it sets it back to: org.gnome.desktop.interface color-scheme = 'default' My dad was an aviator, so I got to hear a lot of exciting words growing up, such as 'civil twilight', which always makes me think of Romeo and Juliet. Sunset turns out to be a surprisingly slippery concept, and very longitudinally mediated. In London in summer there can be plenty of useful light after the sun has dipped below the horizon, and the desktop does not need to go 'darque' the moment the sun touches the skyline. Nautical and astronomical twilight are too late for an interface preference, and in some places at some times of year they can fail to happen in the normal way at all. Civil twilight is when the centre of the sun is 6 degrees below the horizon and when it really does feel like the world has changed character. Location is awkward too, because civil dawn and dusk need latitude, longitude and date. There's some interesting fallback logic to infrequently get a coarse location (good enough for a city) and then fall back to cached data if available as Night Light already needs much the same information. Frequent readers will remember my concerns over London, Ontario being above London, England in many search boxes so there is no virtue in making the user type London into another small box. If neither source has usable coordinates, nothing changes. Manual override behaviour is another thing that avoids annoyance. If the extension sets dark mode at dusk and I then change GNOME back to light mode, I meant that. After any override, it waits until the next scheduled dawn or dusk transition before touching the setting again. Solar time code has an unreasonable number of edges for something everyone thinks they intuitively understand. Keeping with my aggressive policy on internationalisation the tests keep London as the ordinary case, then poke at time zones, date line longitudes, DST changes, Antarctic stations, Arctic towns, awkward offsets such as Lord Howe and Chatham and cases where civil dawn or dusk may not exist at all. My time reading brr and pretending to be in New Zealand to solve work bugs was not poorly spent. Right now Sunset Appearance can be built from source. At some point I may choose to distribute it more widely, or even see if someone has already solved my problem better.

- GNOME Internationalization & Localization: Some news about the internationalization project (2026/06/02 12:00)
  This first blog post marks the opening of the internationalization blog! The i18n team will use it to share news and projects on the current plans. Don't forget to subscribe! The i18n team has seen some changes recently, at the beginning of 2026 and we thought it was necessary to publicly announce this change and introduce ourselves a bit. Before all, we want to greet and deeply thank all internationalization coordinators that participated in the project so far and made GNOME what is is now. We are a global software community of volunteers, leading the free software ecosystem and are accessible in many languages. With this, we cover almost everyone on Earth. Thank you very much Andre, Alexandre, Claude, Daniel, Gábor, Gil, Mario, Piotr, Petr, Kjartan and all the others. Without you this wouldn't have been possible. What is internationalization? Internationalization, or i18n for short, is the act of ensuring software or documentation can be used in other languages, countries, and cultures. This means designing and developing applications in a way that removes barriers to localization, making it possible to adapt them without

requiring significant engineering changes. In practice, this involves separating user-facing text from the source code, so it can be translated easily, and ensuring that the software correctly handles different character sets and writing systems, including right-to-left scripts. It also means being mindful of cultural conventions such as date and time formats, number formatting, currencies, and units of measurement. Internationalization goes beyond text. It includes accommodating differences in sorting rules (collation), keyboard input methods, plural forms, and even layout considerations, as translated text can vary significantly in length. Developers must also ensure that their software supports Unicode and uses libraries or frameworks that simplify handling these variations. For the GNOME community, internationalization is a collaborative effort between developers, designers, and translators. By preparing software properly, the i18n team enables localization contributors to focus on producing high-quality translations, ensuring that GNOME is accessible and welcoming to users all around the world. A new team The team has reborn with new faces: Anders Jonsson, Rafael Fontenelle and Guillaume Bernard, respectively coordinators of the Swedish, Brazilian Portuguese and French team. Let’s introduce ourselves a bit… Rafael (@rafaelff) is coordinator of the GNOME Brazilian Portuguese Team for more than 13 years after a short but intense period of contribution as translator. Besides GNOME, he contributes to the translation of Python Docs, R language, Fedora, TranslationProject (GNU projects, etc.) and others. Also maintains some packages in Arch Linux’s AUR. Anders (@ajonsson) is coordinator of the GNOME Swedish Team for over 10 years, translator in the Swedish branch of the Translation Project, and a member of the GIMP Team with a focus on internationalization questions and testing. Guillaume (@gbernard) is coordinator of the GNOME French Team since this year after 14 years of contributions, first as a translator, reviewer and after a few years, he has been involved in submitting team’s translations. He is the maintainer of GNOME Damnes Lies, our translation platform since 2020. He took this responsibility after years of dedication from Claude. Thank you again for this mentorship!

- Steven Deobald: Stay and fight. (2026/06/01 01:59)
  Nine months ago, I had to field quite a few angry comments from folks who told me they intended to drop their GNOME Foundation memberships in the wake of confusing and opaque board behaviour. I say to you now what I told each of them back in September: Stay and fight. The GNOME Foundation saw a much needed — and long overdue — changing of the guard back in August of 2025. In the past 12 months, the Foundation has finally made the improvements it should have been making over the past decade: 2025-05-09 – GNOME’s infra team had its first management review — it was the cleanest infrastructure review I’ve performed in my career. [edit] 2025-05-16 – Foundation Reports begin in earnest as a first small step toward a transparent GNOME Foundation. We begin the hunt for a new Treasurer. 2025-05-23 – We started a Foundation Handbook to match handbook.gnome.org. (This has since migrated to a wiki.) We started moving all the Foundation’s documents into a central location. Project management began at the Foundation for the first time ever. 2025-05-28 – The Foundation publicly acknowledged that attacks on our Matrix servers, using illegal images, constitute crimes. 2025-06-06 – Both donate.gnome.org and (later) fellowship.gnome.org are pitched and accepted by the board. We brought on Deepa Venkatraman as Treasurer. Bart Piotrowski set up vault.gnome.org for passwords. 2025-06-14 – Andrea Veri completed the transition to donated AWS resources for GNOME infra. 2025-06-20 – donate.gnome.org is released, thanks to the hard work of Bart, Sam Hewitt, and Jakub Steiner. 2025-06-26 – The “Donate Less” campaign begins, in anticipation of the outbound program that would become fellowship.gnome.org. 2025-07-05 – The concept of fellowship.gnome.org goes public. Work on the corresponding donate.gnome.org shell notification starts. We tightened fiscal controls. We added redundancy to all our financial, legal, and operational processes. We interviewed a pipeline of candidates and selected Ignacy Kuchciński to complete the work under the Digital Wellbeing grant. 2025-07-12 – We invited postmarketOS to the Advisory Board. 2025-07-21 – We started stabilizing the GNOME Foundation’s finances for the long

term by redefining the Board Reserve and taking a hard look at balancing year-on-year (annual recurring) revenue and expenses. We added the first-ever redundant signatories on bank accounts. 2025-08-08 – We created a shared online space for Advisory Board members to collaborate. 2025-09-05 – First corporate sponsor. 2025-09-12 – Deepa's budget process is "the best the Foundation has ever had," according to multiple directors. 2025-10-10 – Digital Wellbeing is delivered. The Foundation gets a much-needed credit card policy. 2025-10-24 – A new Finance Advisor arrives. (An important role at a 501c3.) 2025-11-28 – The budget is balanced. More importantly, the budget report contains the commitment to balancing recurring expenses and recurring revenue, continuously. 2025-12-19 – Deepa joins as a full director and remains Treasurer. 2026-01-09 – A new automated accounts payable and accounts receivable system is installed. 2026-03-20 – Financial reporting moves from quarterly to monthly. 2026-04-17 – The Fellowship Program begins! Users' donations come full-circle: a percentage of every donation now goes directly to developers. 2026-05-15 – Finances are on-target. The Foundation opens a position for Finance Director. 2026-05-29 – Four old finance platforms are retired as the finances of the Foundation are automated and simplified. The Foundation introduces a Concern Escalation Policy: if members feel that directors or staff are abusing their positions with policy violations, illegal activity, discrimination, or conflicted behaviour, they're provided the reassurance that they can blow the whistle without risk of retaliation. That's a lot for one little nonprofit. But this is the beginning of GNOME Foundation 2.0, not the end. The work must continue and there is still plenty to be done. If you let your membership expire in recent years, get it back. If you are thinking of leaving, don't. And if you are thinking of running for board elections, run. The GNOME Foundation is the healthiest it's ever been. It's reducing costs and focusing on its actual mission: GNOME. The excellence demanded of GNOME hackers is now demanded of the Foundation, too. You can be a part of continuing that trajectory. There has never been a more meaningful time to join the GNOME Foundation board. EDIT: "GNOME's infra team had its first management review — it was the cleanest infrastructure review I've performed in my career." was previously "GNOME's infra team had its first management review — it was spotless." Someone on Reddit took issue with the use of the word "spotless". This edit serves to accommodate them.

- Gedit Technology: B2B Services around gedit and libgedit (2026/05/30 10:00)
  This article is also available in the B2B Services section on the gedit website. Several business-to-business services are possible around gedit: Development of a new plugin. Development of a new text editor or Integrated Development Environment (IDE) based on the libgedit. Code maintenance. Training. Support. Creation of Long-Term Support (LTS) versions. Developer Experience (DX) guidance. […] Come with your own ideas to collaborate with the gedit project. Target audience Operating System / Linux distributions: for your installed-by-default text editor. GTK-based desktop environments: to maintain a text editor or IDE, and to adhere to your Human Interface Guidelines (HIG). Scientific sector: to build developer tools. Education sector: to build easy-to-learn developer tools. New programming languages / development platforms: you're designing and implementing a new programming language, and you need developer tools for your users. Older programming languages users: you're a big organization and you have a lot of legacy code. You want better developer tools to be more productive. Markup or domain-specific languages: to better promote your language, you would like first-class support for it. The libgedit shared libraries gedit is not just a general-purpose text editor application, there is a "libgedit" underneath! A lot of gedit features are implemented as re-usable code, as a set of shared libraries. So new apps - text editors and IDEs alike - can be built on top. There is an ongoing effort from the gedit project to make more code re-usable. An example of an IDE based on the libgedit is Enter TeX. The libgedit is in turn based on the very flexible GTK graphical toolkit. GTK 3 or GTK 4 The libgedit currently targets GTK 3. If you want to develop with GTK 4, there is the GtkSourceView library (but it doesn't contain all the libgedit features). Another possibility is to first port the libgedit to GTK 4. The plugin system gedit has a powerful plugin system mechanism, to extend the

application. You can leverage it for prototyping additional features, or as the final solution that requires less efforts than creating a new specialized text editor. You can also combine the best of both approaches: First implement a feature that is based on libgedit. Then wrap your feature in a gedit plugin so it can readily be used. Finally, as an option, create a custom text editor app, re-using the same implementation of your feature(s), integrating everything well together. Other developer tools The text editor part is essential, it is the central feature of an IDE. But other developer tools can be developed as well. For instance, Devhelp can be used for browsing and searching API documentation. Almost all its code is re-usable; like for gedit, there is a libdevhelp toolkit under the hood. Advice to not start from scratch A little advice: please don't create a new text editor or IDE from scratch, base your work on existing, high-level libraries like the libgedit. Even if it looks simple on paper, developing a feature-full text editor is a lot of work. Use the libgedit from your preferred programming language libgedit and GTK can be used from a wide range of programming languages, and the support for additional languages can be implemented too. This is thanks to GObject Introspection. See the list of language bindings for the GTK project. Open-source or proprietary software libgedit and GTK are licensed under the GNU Lesser General Public License (LGPL), which allows to develop proprietary software on top. gedit plugins need to be distributed as free/libre software, under the GPL license. Who to collaborate with Sébastien Wilmet, the current maintainer and main developer of gedit and libgedit. You? If you're or work for a consultancy company specialized in GNOME, GLib or GTK, and want to be part of this project, don't hesitate to get in touch!

- Allan Day: GNOME Foundation Update, 2026-05-29 (2026/05/29 16:41)
  Welcome to another update about everything that’s been happening at the GNOME Foundation. As has become my custom, this post covers a two week period, this time from 18 May until today, 29 May. As usual the Foundation continues to be busy, with events, infra, governance, and accounting activities all happening simultaneously. Read on for more information! Events Linux App Summit (LAS) 2026 was held in Berlin over the 16-17 May weekend. I’ve heard quite a few reports now, and everyone seemed extremely positive about the event. Kristi wrote a nice summary if you want more details. The GNOME Foundation had two team members on the ground helping with running the event, which we co-organize with KDE. I’d like to take this opportunity to give a big thank you to the event’s sponsors: openSUSE, Tuxedo, Nextcould and Codethink. This event wouldn’t be possible without your support. In addition to LAS, work is continuing on arrangements for GUADEC 2026. The deadline for travel sponsorship applications has now passed, and the Travel Committee has met to decide who will be funded. Notifications will be going out soon. Board Elections The process is officially underway for this year’s Board elections. Terms on our Board of Directors are two years in length, and each year half the board seats are open for election. This year we have five seats being contested. The 2026 election has a slightly different schedule to previous years. In the past, there was no gap between the candidacy period, in which people can announce their intention to run, and the voting period. This meant that there was little opportunity for last-minute candidates to participate in discussion prior to voting taking place. To address this, we’ve added a one week discussion period to the schedule, which will run between 8 and 15 June, between the candidacy and voting periods. This will hopefully give us opportunity to have more structured and inclusive debate amongst the candidates. We are still figuring out what that might look like, so if people have ideas or want to help, let me know in the comments. GNOME Fellowship We are currently in the very final stages of confirming and announcing the successful candidates for the inaugural round of the Foundation’s Fellowship program. Expect an announcement very soon. Got a Concern? Last week we introduced a new policy for handling of concerns about the Foundation, which is now part of the project handbook. The new policy covers how to report concerns about people who are working for the Foundation, either in a paid or voluntary capacity. It also covers more general concerns about the Foundation. The main goals of the policy are to: have a documented reporting procedure for those who have concerns relating to the Foundation clarify how concerns will be responded to provide reassurance for

those reporting concerns, including that concern reports are welcome, are taken seriously, and will never result in retaliation We hope that this policy will make it clear how you can inform us of a concern if you have one. We also want to emphasise that we want to hear concerns, so we can address them. Please do use the new reporting procedure. Finance/Accounting Work has continued on the finance and accounting operation over the past two weeks. Highlights include: Our transition to a monthly rather than quarterly close reached a significant milestone this week, with the completion of our April finance reports within three weeks of the previous month end. This is probably the fastest ever turnaround for our finance operation, and is a huge win for us in being able to effectively manage our finances. Following input from the board, corrections have now been sent to the accountants for our audit and annual tax filing. Applications are still open for our Director of Finance and Operations part-time contract. Candidates have until 4 June to submit. Finally, as I mentioned in my last update, we are in the process of retiring a number of finance platforms as we consolidate and streamline our operation. This week saw another platform retired, which brings the total number of eliminated platforms to four. Infrastructure Our infrastructure experienced a DDoS attack last weekend, which Bart and Andrea have been dealing with. Thankfully it seems that services weren't too badly affected, and we've already improved our protection against similar attacks in the future. Also on the infra side, Bart wasn't at LAS this year, but he did spend some time writing two great posts about Flathub's internals: How does Flathub even work? and Why are Flathub downloads so slow sometimes?. They're a fascinating read if you're interested in Flathub. That's it from me! As always, thanks for reading, and see you in two weeks.

- Sophie Herold: Updates from the Circle Committee (2026/05/29 14:15)
  We want to share some updates and future plans from the GNOME Circle Committee with you. The Circle Committee is responsible for reviewing and accepting apps and other components into GNOME Circle as well as maintaining the review criteria. The biggest issue for us, and for maintainers who submitted apps, has been the considerable backlog of unreviewed apps. There are cases where apps have been waiting for feedback for years. We are very sorry for those who have been affected by this. We are trying to finally address this issue now. AI Policy As a precaution, we have adopted the AI policy that already applies to GNOME Shell extensions. With the rise of low-quality machine-generated software, there is a risk that GNOME Circle could be overwhelmed by new submissions of this nature, further clogging the review queue. This policy is a starting point, and for now, it will only be enforced on new submissions. We are open to receiving feedback on the AI policy from existing Circle maintainers. In a quick survey, 62% of Circle maintainers reported not using LLMs at all, 34% reported using them for small questions or snippets, and only 3% are taking larger parts of code from them. No maintainer selected the option for no longer writing code themselves. Meanwhile, Flathub has also tightened their AI policy. Since GNOME Circle apps are usually expected to be published on Flathub, this policy indirectly applies as well. Closing New Submissions To make sure that we are focusing on submissions that haven't been addressed for a long time, we will stop accepting new submissions for now. We will reopen submissions when we have made a considerable dent in our backlog. New Handling of Issues This is mostly a technical detail, but from now on, we will close issues that are awaiting feedback from the maintainer. When providing an answer, maintainers should just reopen the issue. This gives us a better overview of which issues are awaiting further processing by us. Early Reminder About the Use of Outdated SDKs Every GNOME release cycle, we spend a considerable amount of time on getting GNOME Circle apps to update their SDK on Flathub before the SDK becomes unmaintained. To avoid this in the future, we will try something new: We will remind maintainers much earlier that they are using an outdated SDK – months before it reaches EOL, rather than just a few weeks. This means apps using the GNOME 49 SDK will soon be part of an automatically generated reminder issue. To get the reminder, please make sure that you have provided a GNOME GitLab account in your project's .doap file. Growing Benefits We are steadily growing the

benefits for GNOME Circle apps and their maintainers. Among the benefits that were added are: Having a student work on your project for Google Summer of Code or Outreachy. Hosting your help pages on help.gnome.org. Getting a tag for your project on GNOME's Discourse. Of course, all contributors and maintainers of GNOME Circle projects are still eligible for GNOME Foundation membership. You can check our full list of benefits. Call for Reviewers We are in constant need of more reviewers in the Circle Committee. If you have already reached out to us, and we haven't gotten back to you, please give it another shot. You can find us in #circle:gnome.org.

- Gedit Technology: News, April-May 2026 (2026/05/29 10:00)
  Here are the noteworthy news about the gedit and Enter TeX text editors. (Some sections are a bit technical). A single package for gedit and its core plugins Before, users needed to remember to install the gedit-plugins package in order to benefit from additional plugins such as Word Completion or Bookmarks. Now, all core - or "official" - plugins are part of the main gedit module. So, Word Completion, Bookmarks and others have been copied into the gedit module, and gedit-plugins is now empty/archived. Note that there are also third-party plugins which are available in other repositories. New version on the Microsoft Store gedit 50.0 is now also available on Windows. The re-implementation of document loading continues It's now a recurrent topic. Progress has been made, but more work is necessary to fully replace the GtkSourceFileLoader internals. This time, here is a general description of the methodologies used along with the current state of affairs, without entering into too much technical details. If you're an experienced developer, it'll most probably ring a bell to you :-) The typical "divide and conquer" methodology is followed: the problem is divided into several smaller ones. The sub-problems are solved individually, providing utility functions and small classes. Then things are built on top, with several layers, gradually solving bigger and bigger problems. It all looks nice, except that an iterative process is also needed, to revise the solutions already built because of unforeseen problems (it happens all the time in computer science). The current progress status is that some utilities need to be reworked in order to be a better fit for the above components, and also to fix a couple of unforeseen problems. Thankfully it's not a "back to the drawing board" situation, as the utilities that lie at the bottom layers don't need any change. For the curious reader, here are the two unforeseen problems encountered: iconv() can output nul bytes even for UTF-8 as the target charset. Embedded nuls are not valid UTF-8 according to g_utf8_validate(). So an additional pass of validation is necessary, to escape invalid UTF-8 bytes. With the new implementation, at most two copies of the whole file content needs to be present in memory. I stumbled across a corner case where three copies would be necessary, so it needs to be fixed. The three copies were: (1) the content containing a single chunk (or at least a big chunk) of invalid bytes, (2) the invalid chunk escaped, and (3) the content as inserted into the GtkTextBuffer. It's just a matter of escaping the invalid bytes directly / at a different place. (Note that this is a concern only for the work-in-progress branch; current stable versions of gedit doesn't suffer from that issue). Enter TeX goodness The Projects feature has been re-implemented in a better way (mostly under-the-hood changes). The module is again on GNOME Damned Lies, so there are many translation updates. Thank you, all translators around the globe! In the past I contributed to the French translation, and I directly saw that GNOME translation teams do very high quality work. So, thanks again :) ! Other stuff As usual there are some other work items fixed or where progress has been made. Among others: The comment/uncomment actions have been fine-tuned. A prototype has been created in order to improve the main "sandwich" menu in gedit. Code re-use and Business-to-Business opportunities There is a lot of potential with the libgedit modules (or also with GtkSourceView for GTK 4), to grow the project into something like the GStreamer multimedia project (but at a smaller scale) where there is a lot of B2B activity with consultancy companies helping to create new products for other businesses. Especially with libgedit-tepl, the name "Text editor product line" means that it's possible to create other text editors or IDEs products (more easily). An example that I have in mind is Arduino and its specialized IDE. But there are new

programming languages or development platforms that pop up regularly, so there are opportunities to collaborate with them and develop great developer tools for them. I'll talk about it more in the future, but if you read this and are interested to develop such a business model for gedit and/or GtkSourceView, talk to me!

- Laureen Caliman: My Current MR and GSoC Project Start (2026/05/29 00:28)
  Ongoing Work Back in March, I started to tackle this MR for GNOME Crosswords. It allowed me to learn a lot about navigating the codebase, adhere to naming conventions, and collaborating with others involved in Crosswords. Crosswords Editor features a section for users to input metadata, such as author name, puzzle URL, and date. Originally, the MR was to convert and store the user's manually typed date to ISO8601. This means storing the entered date in the form YYYY-MM-DD, while the user just needs to type in a date using the GDate struct. Upon code review, the scope expanded to move from manual typing to selecting a date on a drop-down GTK calendar widget. I was able to implement this concept by following the architecture of an existing widget (edit-shapebg.c/h/blp) which placed shapes on the crossword puzzle. Then I connected the new calendar widget to the metadata so the chosen date from the now drop-down calendar will be stored in ISO8601 format yet display respective to the locale of the person using g_date_strftime. Project Start With the start of GSoC, I have spent a lot of time this week reading, white-boarding, and began setting up some structure to adding vocab-style crosswords to the libipuz library. The basic constraints and workflow that I mapped are: A user can input up to 30 words (strings), with a 25 character capacity. Upon hitting enter or a button to generate, a Depth-First Search (DFS) algorithm starts to piece the words together on a blank grid. The algorithm can backtrack to tear apart words and rearrange them until a valid graph is formed I'm going with DFS at the moment to prioritize the longest-to-shortest word seeding the graph in order to increase possibilities of connections Every time the user generates the graph, it will be treated like its own isolated creation, recalculating the layout from scratch to allow for cleaner undo/redo states and easier addition/deletion of words I don't want words being placed right next to each other unless it was intentional for a 2+ word answer, which each word would be separated by thick lines Words should only connect through intersections The frontend does not mutate the grid but will be able to pass actions to the backend and return a new state For instance: say a user starts off their grid by typing "CAT" and then "DOG." These words don't have common characters, so they cannot be connected. However, instead of erasing the user's desire to incorporate "DOG," the string gets tucked away and saved in a GArray Now our user types and enters "CADET." The UI takes our list and passes it to the algorithm to start chewing on and deal onto the board anew. This time, allowing for "CAT," "DOG," and "CADET" to appear, without the user having to retype "DOG" I went back and forth with myself on how to represent the puzzle's state in memory and disk. To write a custom GObject, or not. The Argument Against: Standard IpuzCrossword objects have a handle on grids and JSON saving, and one could write a new function that takes an array of strings and return a standard IpuzCrossword. The Argument For: Revisiting the "CAT," "DOG," and "CADET" conundrum; say an educator is creating a puzzle for his/her/their class, and they have just one word out of 20 that don't connect to current existing possibilities of intersecting words. If they save their puzzle, it transforms to a standard IpuzCrossword, and the floating word are permanently abandoned. At the current moment, I want the puzzle to be its own vocab-style workspace. Therefore this would allow for the floating word to instead sit in the sidebar and wait for further instruction for a future inserted word that works, or to get deleted. Of course, as I gain more insight about the library, states, and optimization within the codebase, this approach is subject to refinement.
- Benjamin Otte: Snapping (2026/05/28 01:41)
  With the release of 4.23.1, GTK's renderer will come with a new feature that we've called snapping. How does it work? Snapping is enabled by

calling gtk_snapshot_set_snap(). If enabled, it will slightly adjust the placement of rectangles when drawing so that they align with the pixel grid and don’t cover half a pixel. Content drawn with GTK is scaled automatically by the desktop’s scale factor. But with the arrival of native fractional scaling, it is no longer possible to know if content is aligned to the pixel grid. While that is usually not a problem, there are a few cases where it is: Sprite grids Gameeky is a learning game that plays on a grid. Unfortunately, on a fractionally scaled machine, it can end up looking like this: Once those sprites are snapped to the pixel grid by rounding to the nearest pixel, the same image looks like this Sharp images Often Applications want to display images in a way that matches the pixels of the image 1:1 with pixels of the monitor. This is a challenge on a fractionally scaled display. Not only is it important to get the scale factor right, it’s also important to align the pixels correctly, or they will appear slightly blurry. The use case is not just image viewers that want to offer a 1:1 zoom factor, but all applications that redirect drawing, from game emulators to viewers like Boxes or Connections. Hardware optimizations And finally, there are optimizations like graphics offload that rely on content being aligned to the pixel grid or the hardware cannot optimize them. So it is important to snap content to the pixel grid for those cases. Why don’t we just always snap to the grid? There is one big problem with automatic snapping: smoothness. Because snapping only works on full pixels, doing slow animations causes content to jump from one pixel to the next. And that causes jitter. The main situation where one can see this is smooth scrolling, like in this example: https://blogs.gnome.org/gtk/files/2026/04/jitter.webm Summary The next GTK release will offer a new way to tame the effects of fractional scaling. Please try it out and let us know how it works!

- Michael Calabrese: Synchronizing Timeline Ticks with GES Framerates in Rust (2026/05/27 00:00)
  While working on my GSoC project (rewriting the Pitivi timeline in Rust), I ran into an issue getting precise UI ticks that map to the absolute nanosecond timestamps of the video frames. Initially I hardcoded NTSC fractional math (24000/1001) to calculate the boundaries of frames. This led to issues with truncated timestamps, and had a glaring issue with other framerates (like 30fps). I needed a more robust solution that could handle any framerate and provide accurate tick positions. I assumed that I could extract the framerate directly from ges::Timeline, however there is no direct getter in the Rust bindings. After some digging, I discovered that the framerate is actually stored in the gst::Caps of the timeline's video stream as a gst::Fraction. My Approach The steps I used: Enumerate the timeline tracks (timeline.tracks()) Filter for the video track (checking ges::TrackType::VIDEO) Read the track's restriction_caps Extract the gst::Fraction My helper I wrote a helper function to extract the framerate from the timeline: pub fn get_fps(&self, timeline: &ges::Timeline) -> Option<(i128, i128)> { timeline .tracks() .into_iter() .find(|track| track.track_type().contains(ges::TrackType::VIDEO)) .and_then(|track| { let caps = track.restriction_caps().or_else(|| track.caps())?; let structure = caps.structure(0)?; let fps = structure.get::<gst::Fraction>("framerate").ok()?; // Extract the safe numerator and denominator Some((fps.numer() as i128, fps.denom() as i128)) }) } // ... inside the timeline injection logic: if let Some((fps_num, fps_denom)) = self.get_fps(timeline) { self.fps_num.set(fps_num.max(1) as i32); self.fps_denom.set(fps_denom.max(1) as i32); } else { // Default to 23.976 if we can't find a valid framerate caps self.fps_num.set(24_000); self.fps_denom.set(1_001); } I make some assumptions here, such as only one video track existing, and that the framerate is always present in the caps. This solution made the tick spacing and labels line up with the timeline’s actual frame boundaries at any framerate.
- Nick Richards: Fuzzy Time Everywhere (2026/05/26 20:57)
  I do not always want to know what time it is. This is a slightly awkward position for someone who keeps making clocks, but there we are. Quite often the useful answer is not 17:42. It is “quarter to six”, “nearly lunch” or “you should probably start thinking about leaving”. The precise time is useful when catching trains, baking things and joining calls; the rest of the time it can be a bit much. So I have been working on fuzzy time for

a while. The first version I made was for the Pebble, which remains one of those devices that makes later technology feel slightly ashamed of itself. A small always-on screen, good battery life, physical buttons and just enough personality. It’s not tokyoflash after all. The current versions are Fuzzy Time GB, a Wear OS watch face, and Fuzzy Clock GB, a GNOME Shell extension. The Android version is quite a funny object internally. It is a Watch Face Format v2 face, so the APK has no app code: android:hasCode="false" The face itself is declarative XML. Since writing thirty-six thousand lines of watch face XML by hand would be a cry for help, there is a generator which writes the cases out from the same fuzzy time rules. For every hour and every five-minute bucket it emits the condition, text and separate interactive and ambient versions. That sounds excessive until you look at the details; and then it still sounds excessive. There are lots of pernickety things that give this the correct GB locale to my ears. “Five Past Midnight” is a real phrase. 23:58 should say “Midnight”, and if the date is visible it should be tomorrow’s date. 11:58 should say “Noon”. “O’Clock” wants different spacing and weight from “Twenty-five To”. Ambient mode wants smaller, quieter text. A round watch face leaves less room than you think it does. The watch face has a few small choices rather than a settings cathedral: warm white, cool white, soft green, dim amber; system font or Arvo; optional radial complication slots above and below the text. The range complications are deliberately arcs around the edge rather than little widgets in the middle. They can show useful things, but they should not make the face stop being mostly words and calm black space. The GNOME version is the same idea on a different surface. It finds the existing clock label, listens to the same wall clock, respects the existing “show date” and “show weekday” settings, and changes the text. I have wanted to build something like this for years, partly because of Emmanuele Bassi’s word clock extension. That extension was great, but not quite the thing I wanted, so eventually I got around to making my own. One of the few design decisions left that I helped on in main GNOME (which is much better now) is that the shutdown and logout dialogue only updates its timing every so often. It could update every second; the computer is quite capable of counting. But it’s much more pleasant when the number doesn’t twitch constantly while you are trying to decide whether you meant to press the button. You can build both projects from source. I may choose to distribute them in a more structured fashion in future. The Android one is a minimal Wear OS watch face, and the GNOME one is a normal Shell extension that currently supports GNOME Shell 45 to 50.

- Shivam: Journey Starts : Gitg Port to GTK4 (2026/05/25 17:32)
  About Me Hello Everyone! I am Shivam, I am currently pursing my engineering in Electronics. I have been selected for GSoC 2026 for the port of GNOME-Gitg from GTK 3 to GTK 4. I am starting this blog in order to document my journey of porting Gitg. I have been contributing in GNOME from several months and in awe with the supportive and helpful nature of the community. Project As many of you probably know, Gitg is still using GTK3, which means it misses out on a lot of the improvements and features that came with GTK4. The main goal of this project is to port Gitg from GTK3 to GTK4 and then gradually modernize the application. The scope of the project itself is quite large, and that’s honestly one of the most exciting parts for me. Working on this port will help me understand the application interacts with different libraries and components behind the scenes. At the same time, I hope that this work will help the new contributors like me easier to get started contributing to the various GNOME projects Conclusive Goal The final goal of this project is to get Gitg building and running completely with GTK4 dependencies. At the moment, the application still fails to compile, which is expected since many GTK3 APIs are still present throughout the codebase.A separate GTK4 branch already exists where parts of the migration work have been started, and several components have already been adapted to GTK4. This project will continue building on top of that existing effort and gradually move the remaining parts of the application to the newer toolkit. I would also like to sincerely thank the contributor(s) who have worked on the GTK4 porting work earlier. Their efforts created the foundation for this project, and I’ll be continuing from the work they have already done. Thank You For Reading! PS:- I would also like to thank Alberto Fanjul for

mentoring me in this project and Felipe Borges for this time and support.

- Thibault Martin: I realized that A cheap VPS is a good front (2026/05/22 16:00)
  I have a server at home. It runs a Kubernetes cluster and a few services. I want to expose them to the Internet, so I can e.g. share public links from my Nextcloud, or synchronize my Kobo reader with Grimmory. But I don't want to expose my home IP to the world, and I want to have some reasonable protection against unsophisticated DoS attacks. I realized that I can achieve that with a cheap VPS that acts as a front, HAProxy, and Wireguard. I rented a tiny VPS for €4/month at Infrawire (1 vCPU, 2 GB RAM, 25 GB NVMe). I installed a Debian 13 on it, because I want that front server to be as stable and low maintenance as possible, and installed the Debian-packaged HAProxy onto it. I also installed Wireguard. The VPS has a publicly accessible IP, so it will be my Wireguard server: my server at home can reach the VPS to establish a tunnel, the opposite is not true. On my k3s node, I've installed Wireguard as well. I configured Wireguard on the VPS and my k3s node to establish a tunnel between the two. I've also bound the sshd on my VPS to the wireguard address. Infrawire offers a console so I can unstick myself if I locked me out of my own server (e.g. by misconfiguring Wireguard on any side, or if my server at home had any failure). I pointed all my DNS records to the VPS. The HAProxy is a "dumb" tcp forwarder, so I can keep operating like before on my cluster. In particular, HAProxy doesn't do TLS termination. My certificates are fetched on my cluster by cert-manager like before, using the http-01 challenge and Let's Encrypt. I could also move to dns-01 challenges, but http-01 just works and lets me switch to a registrar without an API if need be. That way, I don't need a fixed IP at home, and I don't have to do any port-forwarding from my home router to my k3s cluster. Even better: the VPS has an anti-DDoS protection included, and I can also configure HAProxy to refuse too many connections from a same IP, I can make it close TCP connections that take too long to establish, and more. If my VPS gets hammered, I can still access my services from within my home network.
- Michael Catanzaro: Single-Click Code Execution Exploit for Evince, Atril, and Xreader (2026/05/21 20:49)
  CVE-2026-46529 is an argument injection vulnerability in Evince, Atril, and Xreader caused by missing shell quoting when composing a command line. The reporter, João Medeiros, has published a GitHub repo for the CVE and a blog post with the story of how he discovered the flaw and developed the exploit. He also created an Atril security advisory and an Evince issue report. The vulnerability is fixed in: Evince 48.4 (fix commit) (I originally reported that it is fixed in 48.2, but there was no successful release for that tag) Atril 1.28.4 and 1.26.3 (fix commit) Xreader 4.6.4 and 3.6.7 (fix commit) If you use one of these PDF readers, update immediately. Or at least please be seriously paranoid about clicking on links in PDFs until you do update. This vulnerability also affects Papers, but it’s probably not urgent to update Papers. (No, not because it uses Rust. Keep reading!) The Flatpak sandbox could have drastically reduced the danger of this attack, limiting the compromise to only files that you had previously opened in the PDF reader. Sadly, Evince and Papers both use sandbox holes that render the sandbox totally meaningless. (Atril and Xreader are not available on Flathub.) The Vulnerability When you click on a link in a PDF, Evince may execute itself to display the link. Normally the command line used would look something like this: /usr/bin/evince --named-dest=/home/foo/hello.pdf But an evil PDF may trick Evince into executing a command that is quite different than expected: /usr/bin/evince --named-dest= --gtk-module=/home/foo/evil.so /home/foo/hello.pdf Oops. The first part of the command is always going to be /usr/bin/evince, but the evil PDF is nevertheless able to unexpectedly load a GTK module into Evince. The fix is to quote the untrusted input using g_shell_quote() to ensure it cannot “break out” of its intended context: /usr/bin/evince --named-dest='/home/foo/hello.pdf' Or: /usr/bin/evince --named-dest=' --gtk-module=/home/foo/evil.so /home/foo/hello.pdf' Much better: now the threat is neutralized. g_shell_quote() is safe to use even if the untrusted input itself contains quotes. (However, beware: this only works because GLib is parsing the command line itself, and GLib is not a real Unix shell. It’s not safe if the input is going to be passed to an actual

Unix shell. It might not even be theoretically possible to do that safely, because it’s valid for filenames to contain entirely arbitrary characters!) All GTK 3 apps support the --gtk-module command line argument for injecting a shared library into the application. The library may of course then execute whatever code it wants via its library constructor. But GTK 4 no longer has standard GTK command line flags, so this does not work for GTK 4 applications like Papers. It’s still possible to tell a GTK 4 app to load a GTK module, but only via environment variables, not via command line flags, and I don’t see any opportunity for the malicious command to set environment variables. It’s probably not possible to exploit this vulnerability in Papers: although it has the exact same vulnerability as the other PDF readers, the impact is different. The Exploit So far this looks like a pretty typical security bug. OK, so if you trick the user into downloading an archive (or perhaps a git repo) that contains both a malicious PDF and also a malicious shared library, then you can trick the PDF reader into loading the shared library and thereby execute arbitrary code. That’s a pretty bad foreseeable exploit, sure, but at least the attacker is at considerable risk of arousing suspicion if the user is trying to download a PDF and also receives a shared library. You’d have to try pretty hard to hide the library in a forest of other boring files if you want the attack to look convincing and unsuspicious. Right? Nope. João used Claude Opus 4.7 to develop a sophisticated script for building malicious polyglot PDFs that are simultaneously both valid PDF files and also valid ELF binaries, so the attacker only needs to trick the victim into downloading one evil PDF file. When the victim clicks on a link in that PDF, the PDF reader will dlopen the PDF itself. The PDF/ELF polyglot’s library constructor will then execute arbitrary code. Much less suspicious, and much scarier. Polyglot files are not entirely novel, but I’d still say this required substantial creativity and expertise from the AI, and substantial persistence from the human. Needless to say, very nice job to both Claude and João. You can easily build your own malicious PDF using the provided script and sample GTK module. The script in the Evince and Atril issue reports requires that the attacker predict the absolute path that the malicious PDF file will be saved to; however, João’s blog post and GitHub repo refine the exploit to remove that requirement. Thoughts on AI Vulnerability Reports A human inspecting this code should have been able to find the parameter injection vulnerability, but that requires considerable time and effort, so unsurprisingly nobody did. We’re probably in for a rough time in the short term as the volume of AI-generated vulnerability findings remains temporarily very high and attackers have a much easier time crafting working exploits. But in the long term, I expect we are going to be much more secure than we were before, so this will be worth it. A human working alone would have almost certainly stopped and moved on after finding the vulnerability. Claude allowed taking the investigation much farther. It’s highly unusual for a GNOME vulnerability report to come with a working exploit. This is a dangerous change. Perhaps it will be a one-time event, but I suspect we will be seeing more frequent exploits in the future. Silver lining: the exploit helps us better appreciate the severity of the issue. It’s often hard to assess how bad a vulnerability is. If not for the weaponized exploit, I would have thought this bug was not very scary, and would have treated it as not a big deal. We would have fixed it, perhaps or perhaps not with a CVE ID, surely without any blog post or fanfare, and probably without distro security updates. But since there is an exploit, we instead had no doubt that this vulnerability was dangerous, and were able to handle it accordingly. Several GNOME projects have begun outright prohibiting all AI-generated contributions, including issue reports, with no exception for vulnerability reports. Such policies are misguided and unacceptable. I can sort of understand why some projects might (misguidedly) wish to prohibit AI-generated code contributions. OK, fine. But blocking AI vulnerability reports will make GNOME less safe. AI-assisted vulnerability reporting is the new industry standard for good reason: it is highly effective. Some humans are not good at preparing AI-assisted vulnerability reports and will spam maintainers with low-quality reports. Sometimes they will be outright bogus, although more often there may be valid underlying bugs with exaggerated severity claims or bad proof of concept demos. This is annoying, but bad issue reports are a cost we are just going to have to accept and deal with. The quality level of AI vulnerability reports reviewed

by conscientious humans — as well as AI assessments of AI vulnerability reports — is now often quite encouraging. But just like humans, AIs may also miss things, especially subtle distinctions that may be highly relevant. Although I'm quite impressed with these AIs, we still need experienced humans to review and manage reports. Please don't abuse the technology by submitting vulnerability reports that you do not understand or have not validated. And certainly please do not allow an AI agent to interact with an issue tracker on your behalf! For Security Geeks This was my first time scoring a vulnerability using CVSS 4.0 rather than CVSS 3.1. It's also the first time I wasn't terribly confused about how to set the parameters, because the scoring guide contained answers to all of my questions. Nice. My CVSS vector for CVE-2026-46529 is CVSS:4.0/AV:L/AC:L/AT:N/PR:N/UI:A/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N, the base score is 8.4, and I'm pretty sure my choices for each parameter are good. By comparison, using CVSS 3.1 I came up with CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H and base score 7.8.

- Bart Piotrowski: Why are Flathub downloads so slow sometimes? (2026/05/21 13:11)
  It's probably not your fault. On a cache miss, there are two things a reverse proxy (which Fastly is to us) can do. It can make the client wait until the proxy itself fetches the requested content and then serve it, with subsequent requests being served from the cache. From a user's perspective, it means staring at "hung" process, and people tend not to be understanding when a program is stuck seemingly doing nothing. Instead, the proxy can stream the response from the origin, caching it at the end. This makes the client receive the data right away, although it's not without drawbacks. In a streaming setup like Flathub's, an all-MISS path adds some upstream latency before the first byte, but also limits the download speed to what the slowest link can deliver. As we don't run servers in the same datacenter or on a single backbone network, the hop from Fastly through the caching proxy to the master server incurs a penalty that may affect how quickly the data gets back. In order to cache files larger than 20MB, Fastly expects customers who use streaming misses to use segmented caching. Anything larger than that gets broken down into smaller chunks. When Fastly wants the data from us, it will add a Range header specifying which bytes we should respond with. Fastly will then serve the request after reconstructing the file from various chunks. Our caching proxies also use the value of the Range header in the caching key to avoid requesting the full file over and over again from the master server as well. While great for caching, many concurrent range MISSes can turn what would be a sequential file read into scattered, random reads. It wouldn't matter with SSD or NVMe, but as the repository is stored on HDDs, when combined with streaming misses, it can turn cold transfer speed into min(network bottleneck, ZFS random-read bottleneck). Counterintuitively, you may improve your download speeds by aborting the ongoing Flatpak operation and starting it again. While the initial request was slow, there's a non-zero chance it went through all the caching layers and it will become a cache hit in the meantime. FlatpakLet's talk Flatpak. When installing or upgrading applications, Flatpak will try to use delta files. A typical delta is an update file that contains only the difference between versions. There are also from-scratch deltas, which effectively are an archive with all files required to install an app from scratch, thus the name. Flathub generates a single upgrade delta and a from-scratch delta for the latest version. Delta generation is an expensive process in terms of disk reads and writes, but also disk space. Because our ZFS setup isn't exactly the fastest, generating more delta files also affects how quickly we can publish an update. Yes, in theory we could be doing this out of band but we don't. In hindsight, Titanic wasn't unsinkable after all. What happens if you are not updating often enough? A lot of suffering. Flatpak will download each missing file between the version you are on and the one you want to upgrade to, separately. This is an almost certain cache miss causing even more random seeks on the master server. At this point Flatpak would be better off downloading the from-scratch delta but it can't. The behaviour is controlled by OSTree, which doesn't offer any knobs to affect it. It is the right choice if the goal is to limit the bandwidth used by the client to fetch updates, but an incredibly bad one for anyone on a reliable connection; downloading a single large file is almost always faster than fetching multiple smaller

ones. What do? Some brave soul could fix OSTree to apply a better heuristic on when to use from-scratch deltas for upgrades, or at least make it expose an API that lets Flatpak choose. For the rest of us mere mortals, we can only update regularly or wait patiently for the update to finish.

- Sam Thursfield: Status update, 21st May 2026 (2026/05/21 09:03)
  I often write about how when stuff works well, you take it for granted. It's true for technology: when's the last time you hit a compiler bug in GCC? Once upon a time these were a common thing and you had to choose your C compiler wisely. Yet I haven't recently seen an article that says "GCC is going great" . It's true for people too. When someone does an excellent job maintaining an open source project then, they do occasionally get some gratitude, but — if you do a bad job, it's amazing how quickly the negative comments pile up in the issue tracker, many of which taking subtle or not-so-subtle digs at the project owners. Maybe we created this situation for ourselves by having a prominent "report issue" button but no corresponding "send flowers to the maintainer" button. On that note, a hat tip to Carlos Garnacho for all his work on the Localsearch extractor sandbox which recently got a shout out for its "extremely strong" design. (It's worth noting that Localsearch also stopped using GStreamer to parse media files altogether, which the discussion in that thread missed. We love GStreamer but it isn't the right tool for metadata scanning. The 3.9 and 3.10 series use libav/ffmpeg instead, but given that US software patent laws make it tricky for USA folk to distribute that, the plan is to move to using MediaInfoLib) Fairphone 5 It's coming up to two years since I switched to a Fairphone 5. The real proof of this device will be in 2033 when I manage ten years of using the same phone. Meanwhile, I recently had some issues with it not charging via the USB-C port. I thought it might be a bit tricky to fix, but it really is easy: buy the replacement part (about 20€), take off the back cover, remove a few small screws and switch over the whole USB port + speaker unit. I hear some fellow Android users complaining about Alphabet/Google's intrusive AI integration. Apparently the power button is now the AI button? I use the stock Android, and I know vendors have their hands tied somewhat by Alphabet/Google, so its worth noting that disabling the AI integration on the Fairphone 5 is a single config setting. I'd be interested to know more about the kernel version as it is old as hell. I guess this is a vendor/Android thing, and hopefully most of the many known vulnerabilities in this old version of Linux are mitigated by sandboxing higher up in Android. If you're a high risk cybercrime target then I would definitely not recommend using the vendor Android OS on this device. (Probably best to avoid Android altogether if this is your situation!) So its not perfect, but I just wanted to shout out again that there are some good people doing good work here. If only all smartphones were built like this one. Korg Minilogue XD One reason I'm not writing much about open source software is that I'm spending a lot of my time outside work making music in various guises, these days mainly as part of soon to be huge Galician disco revival group Muaré. This band needs a website, so in future I don't have to link you to Instagram, but you know how the world is at the moment. We do at least have a Bandcamp page. When it comes to music gear, I seem to be a Yamaha guy. It's amazing actually that the same company that made my trombone also makes excellent digital pianos, and if and when I need a motorbike, Yamaha also sells those. When it comes to synths though I've been really enjoying the Korg Minilogue XD. It's cheap, built like a tank and its ten years old so there are plenty of second hand models around. It's not fucking Behringer (please don't give money to Behringer). It's simple and sounds great. But most impressively, it support plugins via a freely available SDK. You can develop your own custom digital oscillators and effects for this thing and deploy them over USB. Of all major pro audio manufacturers, Korg are the only company I know to support this. So even though the hardware is now 10 years old, it can still learn new tricks, and there is an active scene of both free and commercial plugins for the platform. Perhaps the most active commercial outfit is Sinevibes. There is, of course, reddit. The SDK is not truly open source (and few things in pro audio ever are) but it's free from any licensing fees, and the whole thing is sat here in a Git repo. Pretty good. If I'd had more time to prepare I might have a video here of some cool Minilogue XD tunes I made. But I guess you'll have

to wait til next month for that. Until then!

- Richard Hughes: LVFS Sponsorship Announcement: HP (2026/05/20 08:09)
  Some more great news: I’m pleased to announce that HP has also agreed to be premier sponsor for the Linux Vendor Firmware Service (LVFS) as part of our sustainability effort. With the industry support from HP (and our existing sponsors of Lenovo, Dell, Framework, OSFF and of course Linux Foundation and Red Hat) we can turbo-charge the growth of the LVFS even more. Thanks!
- GIMP: GIMP on MS Store now requires Windows Build 20348 (2026/05/19 22:00)
  We received some informal reports in the Microsoft Store reviews telling us that GIMP updates were deleting user data! In particular, plug-ins, themes, brushes, user settings and other add-ons were not being kept during an update from the store. To resolve this, the updated version of GIMP from the Microsoft Store will require Windows 11 or at least Windows Server 2022 (Windows NT Build 10.0.20348.0). Windows 10 is still supported, but only when GIMP is installed from our .exe installer. Note that we are no longer able to support versions of Windows older than Windows 10, and can no longer support 32-bit systems running Windows. Installing the update will prevent future data loss during upgrades. Technical details¶ This was being caused by the way MSIX (the format of apps distributed on Microsoft Store, GIMP included) is designed, which prefers sandboxed user data (like Flatpak and Snap). We have been trying to work with this requirement when running from MSIX, but were not successful. So, to fix this packaging limitation, we were given the special “restricted capability” unvirtualizedResources (specifically virtualization:ExcludedDirectory), to preserve user data on %AppData%\GIMP, just as the .exe installer does. This is similar to what we do on other sandbox distributions such as Flatpak and Snap. After the update, GIMP user data will be copied on upgrades. The update does not affect images or other resources stored outside your GIMP folder. How to migrate the previous user data¶ This restricted capability is only available on Windows NT Build 10.0.20348.0 and later. So, users with older Windows versions will need to use the .exe installer. Note: the binaries inside the .exe installer and the .msix are exactly the same. For users who were using the MS Store version until now and are switching to the .exe installer, here are instructions in order not to lose your configuration data from the MS Store version: Manually locate your config dir at %LOCALAPPDATA%\Packages\ (you may just copy-paste this string into the Windows Explorer). Search a folder starting with GIMP (with random numbers and letters after that) at this location. From there, navigating down to LocalCache\Roaming\ where you will find a folder named GIMP. Copy this GIMP folder into %APPDATA% (again, just copy-paste %APPDATA% into the Windows Explorer to find the correct location). Note that maybe the folder %APPDATA%\GIMP\ already exists, in particular if you also used the .exe installer at some point. If so, it is up to you to decide whether you want to override the data already present with the ones from the MS Store GIMP. This procedure is a one-time thing as a workaround to this exceptional issue. You won’t have to do it again in the future!
- Martin Pitt: Leaving Red Hat (2026/05/18 00:00)
  In December 2016 I left Canonical with one sad and one happy eye, with lots of good memories. Now it’s time to revisit some more! Starting at Red Hat back then was quite a cultural shock, of course. I got used to the new headwear fashion quickly: But never really to the rest of the formal dress attire: The drinking habits I was familiar with, and I quickly learned enough Czech to get “dvě piva prosím”:
- Andy Wingo: soot, solar, sedimentation, sin, & 'centers (2026/05/16 19:29)
  Good evening, friends. Tonight I have a few loosely-knit stories.sootA couple years ago, my house was heated by a condensing gas boiler. It was awful from both an environmental and a geopolitical perspective: environmental, as I would emit somewhere around 2.5 tons of CO2 equivalent per year to heat my home, which compares poorly to the target total CO2e emissions of 2 tons per year per person; and geopolitical, because

although France gets 40% of its gas from Norway, with whom we have no beef, all the rest is a problem in some way. (Algeria, 10%, is the least of my worries; the 20% for Russia and the US respectively are the most, followed by 10% for the Gulf states.)Still, natural gas is better than fuel oil, which we had at my former rental house. It is a lamentably visceral experience to call up the fuel provider and say, yes, s'il vous plaît, can you drive a diesel-powered tanker truck out to my house, unroll the hose, and pour out 1500 liters of toxic fuel oil into a tank under my garden. Yes, I will just burn it all. Sure, see you again next year.Some friends of mine recently had their fuel boiler die, which is itself an experience: one of them came over to visit, completely covered in soot, saying that the chimneysweep (whom he also has to call every year) said that his boiler is on its way out, that the chimney is completely clogged, and now because of the cleaning his basement is also covered in soot; awful. What to replace it with? Apparently despite the prohibition on new fuel-oil boiler installs, it might be possible to just install a new one; or they could hook up to natural gas from the street; or they could install a heat pump. Which to do?To all these questions there is a moral answer, which we can phrase in terms in CO2 emissions and localized PM2.5 pollution, and it is always and everywhere to stop burning things. But fortunately we don't need to rely only on moralism: electrification is just better, in essentially all ways. Owning and operating an electric car is a better experience than a petrol car. Induction stoves are better than gas; I know, I did not believe this for the longest time, but I was wrong. The experience of using a heat pump is pretty much equivalent to gas, so it's a harder sell, but it is a relief to no longer have a pressurized methane tube connected to my house.In the end, I think my neighbors are going to go for the heat pump, despite the 20k€ price tag, labor included. (Oddly, I think the deciding factor was that my neighbor confessed to having had a long chat with an AI chatbot, after which she felt she had a good understanding of the proposed solution and its tradeoffs; make of that what you will!)solarIn late November I got some brave lads to install nineteen solar panels on my roof. Each of these magic rectangles can make up to 500W of power in optimal conditions, but my house faces south, with the roof inclined east and west, so it's unlikely that I will ever hit the full 9.5 kW of potential power.December was… very dark. The panels produced a total of 145 kWh over the month, but I used 1250 kWh of electricity, essentially all to run the heat pump. I live in a basin that is mostly covered by low clouds from November to February, and slanty photons couldn't make much headway through the fog. The house is well-insulated (20-25 cm of wood-fiber exterior insulation on sides, 40 under the roof, though it is an old house with a few less-insulated bits), so it's not that I am leaking lots of heat, and I have a combination of low-temperature floor heating and low-temperature radiators, so it's not that I'm running the heat pump inefficiently to generate a too-high output temperature; it's just, you know, cold in winter. A typical day would be between 1 and 5 degrees C. Cold; cold and dark.Things got a little better in January: 285 kWh produced, though the heating needs are higher than in December, with 1450 kWh total consumed. In February we grew to 419 kWh produced, for 850 kWh consumed. In March we equalized, with about 850 kWh produced and consumed, but although the bulk of my consumption in this month is for heating, the "need" to heat overnight meant that I consume from the grid overnight, but feed in to the grid during the day. I have a small battery (7 kWh), but it's not enough to store the "excess" electricity generated in a day; I should probably arrange to have the system heat only during the day in these months, to avoid taking from the grid.With practically no heating needs now, as you can imagine, I am just feeding a lot of excess to the grid. We're halfway through May, just coming through a cold snap (the peasant lore is that we just passed the saints de glace, the date you need to wait for to plant crops that aren't frost-hardy), but still we've produced more than twice as much as we've consumed (550 kWh vs 220 kWh), and essentially all the excess goes to the grid. The 7 kWh battery is quite enough to cover night-time electricity needs.I didn't know before, but often a solar panel installation doesn't work when the grid is down. This is because the inverters that convert the DC from the panels to AC for the house need to match phase with the grid, and if the grid's phase signal is down, they stop. It's also for safety, so that line workers can repair downed lines

without worrying that every house is a live wire. I spent a little extra to install a cutout that allows the house to run in “island mode” if the grid is down. We almost never have that situation here, though, but it seemed prudent that if we were going all-in on electricity, that perhaps we should take precautions.When you buy a solar installation, you can either have little DC/AC inverters attached to the back of each panel (microinverters), or feed DC from all panels wired in series (they call them strings; there may be 2 or 3 of them in a home setup) to a central inverter. I have the latter. The panels happen to be assembled locally by MaviWatt, though surely the cells themselves are from China. My panels are installed on top of the ceramic roof tiles with little clips and an aluminum structure. (It used to be that sometimes panels would replace tiles and become the roof. That’s not done so much any more here.) Installation is, like, 60% of the price of solar. Often you need scaffolding, though my installers just used ladders; perhaps living in the mountains where I am, there are more people used to doing ropes and rock-climbing and such. I don’t think they took as much care of themselves as they should, though.My inverter is made by Huawei (SUN2000), as is my battery and the cutout (“backup”) box. Some batteries have their own microinverter, allowing them to consume and produce AC, but this one is DC, hence the need to have the same brand as the inverter. It sends all my electricity usage data to China or something, so that it can send it to the app on my phone. It’s not ideal from an geopolitical perspective but it is good kit.sedimentationAlthough we haven’t hit the height of summer yet, I would like to offer a few observations that have precipitated out of solution.Firstly, at least in my house, the baseline load without heating is pretty low: 200 or 300 watts or so. (I didn’t know this before looking at Huawei’s app.) We have a recently renovated, not tiny, but otherwise normal sort of house with, you know, the usual lot of modern conveniences, idle chargers plugged in here and there, and also my work computers and such, and it all runs on less than a handful of the old 60W bulbs. That’s interesting.As far as actual load, there are only a few things that count: heating, when it’s cold; it can easily average 2 kW on a cold day. Plug in the electric car (I don’t have a wall box yet, just with the mains plug), that’s another kilowatt. I hardly drive, though, so it’s not a huge load. Using hot water is perhaps the most surprising thing: it can cause a spike up to 6 kW, over a short time, despite the heat coming from the heat pump; probably there is some tuning to do there. The oven and stove are little tiny blips. There’s the kettle, but it’s also a little blip. Nothing else matters: not the dishwasher, not the washing machine, nothing. You can leave the lights on all day and it just doesn’t matter.Call me naïve, but I had hoped that solar would help my electricity usage in winter. This is simply not the case. Though the heat pump is efficient, there does not appear to be a magical energy solution for December, which is the bulk of my energy usage. My electricity bill is fixed-rate: 20 cents per kWh used. Using 4000 kWh or so from the grid over winter costs me 800€; annoying. I don’t have a natural before-and-after experiment as we added on to the house as we were renovating, but for context, in my previous poorly-insulated rental house that was half the size of this one, we’d pay 2000€ or so per year for heating oil. Perhaps I can lower the 800€ via variable-rate metering, to let the battery do some arbitrage, but there are some fundamental constraints that can’t be finagled away.When I got my solar panels, I was resigned to never getting peak power, as they are on two different sections of the roof. It turns out that doesn’t matter: firstly, because 9.5 kW is a lot of power, as you can appreciate from the numbers above. I could never do anything with 9 kW. But secondly, because power isn’t equally valuable at different times of the day: by having east and west roof pitches, I can start producing earlier and continue producing later than if I had, say, a flat roof with panels tilted to the south. And the morning and the evening are the peak hours both for my house and for the grid, so that lets me consume more of my local production both when I need it and when the grid is under higher stress.I was interested to hear that Alec Watson of Technology Connections had reservations about residential rooftop solar. I found a video in which he explains his perspective, which has a delightfully socialist character. His beef is partly due to the net metering scheme in some parts of the US, in which each kWh fed to the grid makes your meter run backwards; Watson finds it unfair, because it lets those wealthy

households who have the capital to install solar to opt out of paying for the grid, which is a social good. In some cases, these households actually capture a part of what consumers pay for the grid, unlike industrial producers who are paid wholesale rates that don’t include transmission. Also, he finds it less efficient overall to install solar panels on houses rather than in bigger solar parks; each euro that society allocates to solar would go farther if we pooled them together.Both points are interesting, but I would offer a couple responses. Firstly, at least in Europe, net metering is not really a thing; we have smart meters and I hear from friends in Portugal that there can even be a charge for grid injection at some times, if the grid is overloaded. France’s case is a bit weirder; I wouldn’t have gotten as large a system as I did, but there was a government program to offer a fixed buyback rate of 7 cents per kWh, stable for 20 years, if you installed more than 9 kW of panels. But given the lack of solar in December, I still pay the grid when I need energy the most.Putting solar panels on roofs is indeed less efficient than putting them on a field. But, we are not in a situation of scarce solar panels: China could make another 350 GW of panels this year if there were demand. An incentive like the 7-cent buyback rate encourages capital allocation to solar, effectively calling these panels into existence. The bank loans me 20k€ at 4%, and the elimination of 3000 kWh that I would have bought from the grid in a year plus the 9000 kWh that I sell to the grid covers the cost entirely, and I get a life insurance policy on the remaining principal. It’s not a great investment financially but it doesn’t cost me anything either.sinAs a person with a conscience, I have always experienced questions of energy as questions of sin; to leave a light on is not simply inefficient but a moral failing. Each kilometer a car travels on fossil fuel carries with it a quantum of guilt and must be justified in some way, otherwise a moral stain attaches.Solar panels and electrification changes all this. 8 or 9 months out of the year, I live in a world of abundance: the electrical generation capacity that I have called into existence is free, clean, and much, much more than I need. Owning and operating a car still has externalities, but the emissions and cost aspects are entirely gone. It’s a funny feeling, and disorienting.I grew up in the south of the US, where everyone has air conditioning. I came to see it as sinful, too; burning things and making emissions just so you could be a bit more comfortable. I haven’t lived in air conditioning since then, but it does get hot in summer, and I would be more comfortable if I could pump heat out of my house. Now I can. I have excess power available right when air conditioning (or, in my case, floor cooling) is needed. On a societal level, solar plus air conditioning is going to be a key part keeping our cities liveable while we ride out higher temperatures.‘centersIt is with a sense of dissonance, then, that I have been experiencing Datacenter Discourse™: there is a lingering language of sin proceeding from an environmentalism born in penury, in a world in which every kilowatt-hour is precious and scarce. If China has unallocated capacity for another 350 GW of panels this year, why stress about a few GW of datacenters?Of course, there are many aspects to these AI datacenters, but today I am just thinking about energy. Given that each GW of datacenter places extra demand on a grid, equivalent to 3 million times my home’s baseline load, or maybe 300 thousand of its winter load, if society wants this kind of datacenter to be a thing, it needs to add that amount of clean energy to the grid, with adequate battery storage to even out supply. We should, as a society, require this via legislation, because the market seems only too happy to use natural gas or even coal if it is marginally cheaper. At least if the datacenter boom busts, we’d be left with more clean energy production.Conversely… and I don’t think I’m going too far here, but causing new fossil generation to come online in 2026, or even prolonging the life of existing generation, should result in the state confiscating all property of those responsible. (I have moderated my previous position, which was hanging.) Such people are not fit to live in society, so society should not allow them to own things.Anyway. I think that those of us that wish “AI” were not a thing are losing the battle, and that we should prepare to fall back to more defensible positions; otherwise we risk a rout. A requirement to bring additional clean capacity online in sufficient amounts should be a baseline ask when it comes to datacenters. We have the productive capacity in the form of solar panels, at an affordable price, more than enough space in terms of the existing cropland that is inefficiently turned into ethanol to burn, batteries

are a thing, and we just lack the political will to turn what could be into what is.And as for AI datacenters themselves: there are enough aspects to argue about as it is. We do ourselves a disservice by weighing down the Discourse with outdated ideas of what is and isn't possible.

- Sjoerd Stendahl: Graphs 2.0 is out! (2026/05/15 19:00)
  After two years of development, Graphs 2.0 is finally out! This will be a shorter blog, as the changelist of the new features have been discussed in the previous post in more detail, you can check this out here in more detail if you're interested: https://blogs.gnome.org/sstendahl/2026/04/14/announcing-the-upcoming-graphs-2-0/ For a quick overview, a quick reprise of the most interesting features can be found here in bullet-point format: New Data Types with proper equations: In this new release, we finally have proper support for equations, where equations are manipulated analytically (e.g. a derivative on $y = 12x^2$ will result in $y = 24x$, and be rendered accordingly), limits are rendered infinitely, and equations can be changed after importing them. To accomodate this change, we now have three different data types. Equations, Imported Data, and Generated Data. Imported Data is regular data that you import from file. Generated Data behaves the same as Imported Data, but you generate the dataset using an equation. For generate data you can also change the equation after the fact, and rerender, or change limits or amount of generated datapoints. New Style Editor: We revamped the style editor. One major change is that you can now easily import new styles based on matplotlib-style themes, and you can also export your style and share it with others. If you have a nice style you want to share with us, open an issue on the GitLab page and let us know. We're looking to expand the default choice of styles :). Furthermore, when editing a style, you finally see a preview of how this affects the canvas. This way you don't need to guess, or go back and forth when finetuning a style. UX-changes: Whilst the general look-and-feel of Graphs is still mostly the same, we did make some UX-changes with how we handle settings. Mainly, instead of showing a modal popup dialog, settings that affect the canvas itself (i.e. item and figure settings) are now shown in the sidebar instead. The reason for this, is that the popup dialog hid the figure that you're editing, making it difficult to see how your changes actually affect your canvas. Improved data import. We revamped the data import completely. First of all, we have made the codebase here much more modular, making it easier to write new parsers for other filetypes. We now added support for SQLite Database Files, Microsoft Excel Sheets and .ods files from LibreOffice Calc. It's also now possible to import multiple files at once, and finetuning the settings for each file individually. Another nice feature here is that you can now import multiple datasets from the same file without having to reimport them. Finally, we added proper-support also for single-column imports where x-data can be generated using your own equation Error bar support: Graphs now has proper support for error bars. The error bar style can be set globally in the new style editor, or individually for each item. Reworked Curve Fitting: The curve fitting logic has almost been completely rewritten. Whilst it mostly still works the same, the confidence band that is shown is now calculated properly using the Delta-method, instead of using a naive way using the limits of the standard deviations. We also added support to show the residuals to verify your fits, and added more useful error messages when things go wrong. The results in the curve fitting dialog now also show the root mean squared error as a second goodness-of-fit figure. The parameter values themselves in the curve fitting dialog are no longer rounded (e.g. 421302 used to be rounded to 421000) and finally custom equations in the curve fitting dialog now have an apply button, greatly improving the smoothness when entering new equations Proper Mobile Support: With this release, we now officially feel confident in stating proper mobile support. The entire UI is tested on real mobile phones using PostMarketOS, and everything works properly. Labels are now also ellipsized earlier on narrow displays, so that the UI remains useable. Reworked Figure Exports: Figure exports are reworked. Instead of simply taking a snapshot of the current canvas, you can set the actual size in pixels when exporting a figure. This is vital when trying to create reproducable figures for e.g. publications. Of course, you can also easily still use the same canvas that you see in the application.

Quality of life changes: We haven’t even gone through each individual change we made in the blog, but here’s a quick fire-round of more quality-of-life changes: It is now possible to have multiple instances of Graphs open at the same time The style editor now also has the option to draw tick labels (so the numeric values) on all axes containing ticks, this is not supported by default in Matplotlib, so we had to add our own parameter here Graphs now inhibits the session when unsaved data is still open, so you get a warning if you close your computer with unsaved data. Added support for base-2 logarithmic scaling Graphical fixes for the drag-drop animations, which used to look somewhat glitchy Panning and zooming are now done consistently on all axes when using multiple axes Data can now be imported by drag-and-drop into Graphs The subtitle now also shows the full file path for Flatpaks Limits can now easily be clicking on the numbers near the axes The custom transformation has gained the following extra variables: x_mean, y_mean, x_median, y_median, x_std, y_std and counts Warnings are now displayed when trying to open a project from a beta version The many code refactors and reimplementations from Python to Vala makes the application more robust, and significantly more performance. Especially when working with larger datasets. This release took a long time to get right, but we’re happy to get the new features to the public. Graphs is handcrafted by human hands, which takes more time than LLM-based slop. But the longer manual process does allow us to think through changes, and make intentional decisions with human care. I am very proud to say we are able to deliver something intentional where we can deliver the polish that both Graphs, as well as the users deserve. As always, thanks to anyone involved which includes everyone who has been providing feedback, reported issues, contributed with code, or helped in any other possible way. And of course especially to Christoph who has been maintaining Graphs with me and is responsible for a large part of the architectural changes that made this release possible. Go get the new release from Flathub here! p.s. On a more personal note with a shameless plug, I will be speaking at GUADEC 2026 about my journey into app development, and how to get into this world as an outsider without a CS degree. Be sure to check that out if you are interested in starting with applications, and want to know how it is to join a project in the GNOME ecosystem, it’s a lot less scary than it may sounds I’ll be joining on-site, so say hi to me there if you have any questions or are up for a chat :). Otherwise the whole event will be livestreamed as well, and you can always reach me at sstendahl@gnome.org.

- Allan Day: GNOME Foundation Update, 2026-05-15 (2026/05/15 16:41)
  Welcome to another GNOME Foundation update post! Today’s installment covers highlights from what’s happened over the past two weeks. LAS 2026 Linux Apps Summit 2026 starts tomorrow! The organizing team, which includes members from both GNOME and KDE, has been hard at work and is on the ground in Berlin making final preparations. The schedule looks great, and it promises to be a well-attended event. The talks are being streamed this year, so make sure to watch our social media for details, and tune in live to hear the talks. GUADEC 2026 Preparations are continuing for July’s GUADEC. The call for Birds of a Feather sessions is currently open. If you want to hold an informal discussion or working session, please fill out the form before 5th June. Applications are still open for travel funding for GUADEC. The deadline for submissions is 24th May – that’s just over one week. Board meeting This week the Board of Directors had its regular meeting for May. A summary: The Board authorized the closure of a bank account which we are no longer using. I gave an update on operations over the past month, and got feedback from the Board Felipe Borges from the Internship Committee joined, to give a report. The Board discussed how we can best support the committee. Deepa gave a finance report, which included numbers from January and February. The main news here was that our finances are running close to what was projected for this year’s budget. The Board discussed the draft of the report from the audit we recently underwent, as well as the draft of our latest annual tax filing. This is a routine part of the Board’s work, as it is required to perform a review before these documents are finalized. Office transitions Our long-running effort to enhance our internal accounting processes has continued over the past two

weeks. A notable development has been the retirement of several finance platforms, which have been effectively replaced by the new payments platform that we adopted in January. This platform reduction will reduce operational complexity, as well as workloads. It is still ongoing – we have an additional two more platforms that are currently in the process of being retired. Another highlight has been the launch of a search for a new member to join our finance and operations team. This is a part-time, contract-based role, which has been shaped in close consultation with Dawn Matlak, who is supporting our finance and accounting operations on a temporary basis, and has already been factored into our budget projections. We are looking for someone at director level who brings substantial nonprofit finance experience — including audit preparation and compliance experience — which reflects how much the Foundation's operational and regulatory requirements have grown, particularly in the run up to and following our audit last March, and will provide in-house expertise which will reduce our reliance on external consultants. You can read the full posting here. Thanks for reading, and see you in two week's time!

- Bart Piotrowski: How does Flathub even work? The CDN and caching layer (2026/05/15 10:25)
  There is one specific way in which the non-corporate open source projects typically document how their infrastructure work: not at all, and Flathub is no different. The full picture likely lives only in my brain, and while it could be sorted out by anyone (especially in this LLM age, yay or nay), why should it only be me thinking at night about all the single points of failure? Like any system that evolved naturally, it's all over the place. It's tempting to tell its history chronologically, but even then, it's difficult to find a good entry point. Instead, this post focuses on what happens when users call flatpak install; later entries will cover the website and, finally, the build infrastructure. Buckle up! CDN, caching proxies, the master serverThe secret of making computers work well is to have them not do anything at all, and that's the story behind serving Flathub's OSTree repository. Content-addressed objects are extremely cacheable as they are immutable, offloading the effort to the CDN provider. When the client connects to dl.flathub.org, you can be certain it hits some layer of cache. Almost all the heavy-lifting is done by Fastly. At the peak, when both EMEA and North America are awake and at computers, 50 million requests per hour are cache hits served by Fastly's infrastructure, with a modest 20 million being misses passed down to our servers. There would be no Flathub without Fastly; Fastly does it completely for free, not even for fake Internet points as we are incredibly bad at highlighting what our sponsors do for us. You can't do enough cache, and so various Fastly servers talk to Fastly-managed shield server which caches the most requested objects to avoid spilling over too much to us. For legit cache misses, the request will be served by one of 8 caching proxies we are running at different VPS providers. We use a consistent hashing director at Fastly which will pick the backend based on the path being requested. In the past, we used a dumb round-robin but as a result, each caching proxy had its own independent copy of the working set, wasting disk space and producing a higher miss rate against the master server. Hashing by URL behaves like one big cache instead of N copies. These days, the caching proxy fleet consists of 3 servers at Mythic Beasts, 2 servers at AWS, another 2 at NetCup and a single server at DigitalOcean. We don't collect overly detailed metrics, but on average, each proxy serves around 1 TB/month back to Fastly and pulls roughly 5 TB/month from origin. With only 100 GB of disk space per proxy against a multi-TB working set, we're not so much caching the long tail as smoothing it. In the ideal world, we would be retaining much more data at this layer, but it's not the world we live in. Each of these servers is running the latest stable Debian release. The requests are served by the usual nginx setup with proxy_cache enabled. There is some custom Lua code for invalidating certain paths after publishing new builds finishes (spoilers!). Vanilla nginx doesn't support the PURGE method, and third-party modules like ngx_cache_purge have not seen any maintenance for over 10 years. In the end, it was more maintainable to write Lua code to calculate the caching key of a URL and then run os.remove to "purge" it from the cache. There's also a systemd timer for refreshing the Fastly IP allowlist. We used to expose these servers publicly, but a vision of everything crumbling down

due to a DDoS attack kept me awake at night so this had to change. On the far end of this setup sits a lonely physical server living in one of the Mythic Beasts' datacenters. This is The Server holding the entire Flathub repo on an equivalent of RAID10 in ZFS world: two 2-disk mirror vdevs on which ZFS stripes data across. There is more nuance to this setup, but the ultimate advantage is that we can tolerate a disk failure in each of the mirrors, while being less taxing to resilver after a swap. The entire reachable data set is around 4TB of data, with the remaining 6TB unused. There will be more about the repository maintenance later on! Ironically, it's the only server running Ubuntu. At the time, it was the easiest way to have support for ZFS readily available. We could re-provision it to Debian, but on the other hand, what for? It works fine that way. It has survived at least 2 major upgrades between LTS-es; if it ain't broke, don't fix it. The master server itself has to be partially public as it's where new builds are being uploaded. It no longer exposes the raw Flathub repository for the same reason caching proxies don't. This is accomplished with Tailscale and a lightweight ACL config ensuring caching proxies can talk only to the HTTP server running on the main repo server and vice versa (for issuing PURGE requests). Yes, all involved parties have public IP addresses assigned so this could technically be pure WireGuard setup but I prefer to make this someone else's concern, especially given how generous Tailscale's free plan is. It's not much, but it's honest work. For how little we have, the file-serving half of Flathub's infrastructure works unreasonably well. Stay tuned for part 2!

- GIMP: GIMP @ Linux App Summit and ADULLACT Congress 2026 (2026/05/14 22:00)
  We have been trying to encourage contributors to be more present on various events, international or local. Here is where you will find someone from the GIMP team in the coming weeks: Linux App Summit 2026¶ The Linux App Summit (LAS) brings the global Linux community together to learn, collaborate, and help grow the Linux application ecosystem. Linux App Summit 2026, May 16-17, 2026 - Berlin, Germany It happens this week-end, from May 16 to 17 in Berlin, Germany, and we will have one person attending, Michael Schumacher, one of our long term contributor, as well as member of our Committee. Unfortunately our project did not submit a talk, but we are still interested to meet more of the desktop software ecosystem contributors and see what’s happening around us! So if you attend too and spot Michael, do not hesitate to go and speak with him. He will likely have Wilber stickers to distribute too! 10th Congrès ADULLACT¶ The Congrès ADULLACT is a conference gathering elected representatives of French local authorities, to discuss Free Software usage in the public sector. 10th Congrès ADULLACT, June 4-5, 2026 - Montpellier, France Jehan, GIMP Maintainer, will be present there to showcase GIMP as a Community, Free Software. Obviously GIMP is already quite massively present in France, but as many Free Software, administrators and users alike may not realize how it is being developed, by whom, why and how. Nor do they know that it is being developed by a major part in Europe and more particularly in France. Since one of the two main topics this year is the digital sovereignty, this is quite a major stake in this context. The event happens from June 4 to 5, 2026, in Montpellier, France. As one can imagine, it is a close event for elected representatives and civil servants only, so if this is your case, we hope you will show up and Jehan will be happy to discuss with you! Jehan’s talk will be on Friday, June 5, 2026, at 14:20 (French time) and he will introduce GIMP as a “Free Software and Community”. We hope you’ll be many to attend! (oh and Jehan as well will have Wilber stickers, even though it may less a selling point in such a conference)
- Nirbheek Chauhan: An Esoteric Type of Memory "Leak" (2026/05/14 21:52)
  A little while ago, my colleague Sebastian started complaining about OOMs caused by Evolution taking up tens of gigabytes of memory. We discussed using sysprof to debug it, but it was too busy a time for Sebastian to set aside a few hours to do that.Funnily enough, the most efficient fix at the time was to buy more RAM, since rust-analyzer was also causing OOM issues.A few weeks went by. Restarting Evolution had become a daily ritual for Sebastian. Then, on a whim, I decided investigating this might be a good test for an LLM.I updated my Evolution git repo, built it,

and started up Claude Code in the source root. This was the only prompt I supplied: Find memory leaks in Evolution, current sourcedir. Particularly leaks that could accumulate over several hours. A colleague has a leak that slowly accumulates memory usage to several GB over the course of a day, requiring a restart of Evolution. That is the main focus, but we can fix other leaks in the process.I wish I was lying, but that was all Claude Code needed to find the problem: Evolution just needed to call malloc_trim(0) from time to time.I refused to believe it at first. I was only convinced when we saw the memory drop after running gdb -p $(pidof evolution) -batch -ex "call malloc_trim(0)" -ex detachThis seems absurd! Doesn't glibc reclaim freed memory from time to time?Yes, it does. It calls sbrk() to do that. However, sbrk() can only reclaim free memory at the top of the heap, since it simply moves the program break downward to do so. malloc_trim(0) calls sbrk() and then also calls madvise(..., MADV_DONTNEED) on the free pages, which allows the kernel to reclaim them.So if you have 10GB of unused memory followed by 4 bytes allocated at the top of the heap, your RSS is >10GB, even if you're using a few hundred megs. Till you call malloc_trim(0).Note that you can only get into this situation if you have hundreds of thousands of small allocs/deallocs happening repeatedly. If your alloc is >128KB, mmap() is used for the allocation, and none of this applies.Coincidentally, GLib's use of GSlice for GObject allocations was masking this issue in the past, but GSlice has been a no-op for some time now (for good reasons). Ideally, Evolution should not be using GObject for such ephemeral objects.Lesson learned: if you have memory usage issues and you suspect fragmentation, try malloc_trim(0) before you go thinking about fancy allocators.

- Toluwaleke Ogundipe: Hello GNOME and GSoC, Again! (2026/05/13 18:17)
  I am delighted to announce that I am returning for Google Summer of Code 2026 to contribute to GNOME once again. Following my work on Crosswords last year, I will be shifting focus to the core of the desktop: Mutter. For what it’s worth, I never left; I’ve been working with Jonathan to improve things and add shiny new features in Crosswords. Mutter serves as the Wayland display server and compositor library for GNOME Shell. Currently, a GPU reset invalidates the EGL context and causes the loss of all allocated GPU memory, resulting in the entire desktop crashing or freezing. My project aims to implement a robust recovery mechanism for GPU resets to prevent these session-ending freezes, under the mentorship of Jonas Ådahl, Robert Mader, and Carlos Garnacho. Leveraging the GL_EXT_robustness extension, I will implement reset detection, context re-creation, and re-upload of essential GPU resources, such as client textures, glyph caches, and background images. This will allow the compositor to resume rendering seamlessly after hardware-level failures. Over the course of the project, I will share updates on the progress of these recovery mechanisms and the challenges of managing state restoration within the compositor. I am very grateful to my mentors, Jonas, Robert, and Carlos, for the opportunity to work on this critical part of the GNOME ecosystem. Also, a big shout-out to Federico, Hans Petter, and Jonathan for their continuous support. I look forward to another productive summer with the community.
- Richard Hughes: LVFS Sponsorship Announcement (2026/05/06 12:13)
  Some great news: I’m pleased to announce that both Dell and Lenovo have agreed to be premier sponsors for the Linux Vendor Firmware Service (LVFS) as part of our new sustainability effort. Over 145 million firmware updates have been deployed now, from over a hundred different vendors to millions of different Linux devices. With the huge industry support from Lenovo and Dell (and our existing sponsors of Framework, OSFF, and of course both the Linux Foundation and Red Hat) we can build this ecosystem stronger and higher than before; we can continue the great work we’ve done long into the future.
- Steven Deobald: Apologies (2026/05/06 03:37)
  I believe accountability can be a challenge in a nonprofit, which only makes it all the more important. In this post, I am holding myself accountable. For the avoidance of doubt, nothing that follows has anything to do with my exit from the GNOME Foundation last August. I owe a

few folks some apologies from my time as Executive Director. I have apologized to most of them individually already, where I could. But I believe that public accountability is the antidote to public frustration and I hope this contributes, in a small way, to the GNOME community moving forward. First off, I sincerely apologize to Jehan Pagès and Christian Hergert. I was curt with both of you last summer and neither of you deserved it. From July 23rd to August 29th I was dealing with significant sleep deprivation but that's no excuse for the way I spoke to either of you. I'm sorry. Next, I apologize to the former Executive Directors and active community members who raised concerns to me. Holly, you warned me. Twice. Many other people tried to share their perspectives. I was too focused on the Foundation's financial situation, and I did not take the time to fully understand what I was hearing from you all. I regret that. Sonny To Sonny Piers: I am sorry. I had a long call with you last June. You told me your complicated story. You seemed hurt — but I didn't believe you. My understanding was incomplete and I did not approach the situation with the care it deserved. I'm sorry I didn't do more to support you. Tobias More than anyone, I want to apologize to Tobias Bernard. Tobias, I am sorry. You gave me many hours of your time, patience, and thoughtfulness. You shared your ideas openly and in good faith, and I didn't always meet that with the same level of openness. In particular, when we discussed Sonny's situation, I did not listen as carefully as I should have. I was too focused on my existing understanding, and I failed to engage with what you were trying to convey. You deserved better from me. Sonny is lucky to have a friend like you. Meta This post reflects only my personal experiences and perspectives. It is not intended to make allegations or factual claims about the conduct of any individual or organization. Until Microsoft goes out of business, a permanent copy of this apology can be found in this gist.