KDE Planet - Latest News

- Making wl_shm fast (2026/05/06 13:25)
  While most new applications use the GPU for rendering to achieve better performance and battery life, there are some new applications and a lot of older applications that still use CPU rendering. More specifically relevant for KDE, while QtQuick is GPU accelerated, QtWidgets uses CPU rendering. With CPU rendering, instead of sharing GPU buffers with the compositor, wl_shm is used to present images. “shm” stands for “shared memory”, and is literally just some system memory allocated by the app and shared with the compositor. Why is it slow? The rendering speed of an application using CPU rendering depends a lot on what the application is doing exactly, but a very large factor is simply the sheer number of pixels and thus bytes it manipulates. With high resolution screens, especially single threaded CPU rendering can get pretty slow. Optimizing the application side isn’t my area of expertise though, and not what I’m primarily interested in as a compositor developer. My main goal is to let the application render at whatever speed it can, and to efficiently transfer the results onto the screen. On the compositor side we can’t normally use shm buffers directly. For the GPU to be able to access the data, we first need to copy it to a different buffer that meets the requirements of the GPU. This copy is often done in two steps: copy the data to a GPU-accessible buffer on the CPU copy that GPU-accessible buffer to another buffer in GPU memory With both OpenGL and Vulkan, that first copy is blocking the main thread until it’s complete. You can offload the copy to a different thread with some additional code, but that would just move the CPU usage, rather than reduce it. The second copy is more acceptable, since the GPU does it asynchronously and more efficiently, but on integrated GPUs, this would still end up copying data from system memory to a different region of system memory, for no good reason. The result of these copies is that on high resolution screens with applications using shm buffers, performance noticeably suffers and CPU usage is much higher than it has any right to be. On my laptop with a still relatively new and high end Ryzen 7840U, I could see the cursor sometimes skip frames when quickly moving it over project files in KDevelop, since KWin’s main thread was being blocked by these texture uploads. Normally that’s not really noticeable, but with the power profile set to “power save”, it felt really sluggish. Vulkan will fix it… right? When you hold a hammer, every problem starts to look like a nail. Since we recently started using Vulkan in KWin to fix some other problems caused by OpenGL’s inadequacies1, I obviously looked for a Vulkan solution first. And lo and behold, VK_EXT_external_memory_host does exist, and it’s perfect for this! Or at least it looked like it would be… The extension allows wrapping a “host pointer” (aka a normal pointer to CPU memory) in a VkBuffer or even VkImage2. With a pretty low amount of new Vulkan code, the GPU could asynchronously copy the VkBuffer to a GPU-local buffer. Unfortunately, the implementation at least on AMD comes with some limitations. Because of potential security issues, pointers to anything associated with a file descriptor (which shm buffers always are) can’t be imported this way by amdgpu. There is also the more recent VK_EXT_host_image_copy for optimizing image uploads, but it would only allow removing the second copy rather than the first, so it’s not exactly what I needed. udmabuf to the rescue udmabuf is a Linux driver that can wrap memfd-allocated memory in a dmabuf. A dmabuf is a handle to GPU memory, and memfd is how shm buffers are usually allocated by Wayland clients… so it’s a perfect fit for what I wanted to do. There’s one caveat to this: In order to be able to create a udmabuf from it, the allocated memory must be a range of memory pages3, so location and size have to be a multiple of the page size. Applications didn’t allocate their buffers with that in mind so far, since there was no benefit to it. Fixing that isn’t difficult though! Assuming one memfd per shm buffer (which at least Qt does),

fulfilling the page size requirement should even be free4. With the udmabuf successfully created, we can wrap it into a VkBuffer and do an asynchronous copy to a GPU-local buffer with Vulkan. However, we can do even better: If the stride5 of the buffer matches the requirements of the driver, we can directly use the udmabuf with the GPU. This stride requirement is a bit more of a tradeoff than the page size one, since some additional memory may need to be allocated as padding at the end of each row in the image. Since most GPUs seem to be fine with a multiple of 256, the amount of “wasted” memory is still pretty low however - for example with a 3841x2160 image, it would be 0.55MB or 1.6% more memory used per buffer. So I added code to KWin to attempt to create a udmabuf for each shm buffer, and then import that into the GPU driver. If it fails, we just fall back to the old upload code, but if it succeeds, we don’t need to do any copies at all. The compositor side didn’t take a lot of code, but the application side was much simpler still. Including a comment explaining the reasoning, it merely took a grand total of 18 changed lines of code. The result With the same example of KDevelop I mentioned before, the cursor is now always completely smooth. In terms of concrete numbers, KWin’s CPU usage while scrolling in KDevelop went from 80-90% on one core down to 20%! These improvements will be in Plasma 6.7 and Qt 6.11.2. I would recommend other toolkits and applications that use shm buffers to make the same changes as I did in Qt, it can make a really noticeable difference. I’ll write a blog post about it once there’s more to talk about ↩ With a Vulkan renderer, the VkImage would mean the second copy could be skipped as well ↩ A page is the smallest chunk of contiguous memory managed by the OS ↩ The kernel allocates in pages, so the amount of memory used should be the same either way ↩ Stride is how many bytes are used by each row of pixels in an image. There can be unused padding after each row, which is included in the stride. ↩

- Introductory Post (2026/05/05 17:00)
  Hello, I am Ojas Maheshwari. I got selected for contributing to the project "Implement Font Subsetting when saving PDF files" for GSoC 2026 at KDE community. This site will have all the official documentation and progress updates on what I did through the whole journey including: My approach and plan. The problems I faced and how I solved them. My thinking process wherever possible. This is an introductory page to see if the site works correctly. Thanks :D
- KeepSecret 1.1 (2026/05/05 11:37)
  KeepSecret is our new password management application, based on SecretService, which works both with our old KWallet infrastructure as well as more modern services such as oo7, KeepassXC and many others. Version 1.1 has now been released. This release Has been focused mainly on small usability papercuts and improved messaging to the user. An important aspect of this release is that is the first one that’s available on flathub, so it’s very easy to install and test now, just hop on Discover to find it.
- Introducing the QML Profiler Skill for Agentic Development (2026/05/05 07:02)
  Instead of a painstaking row-by-row or slow flame graph reviews, the QML profiler skill for agentic development allows developers to delegate code performance profiling to AI agents.The skill guides the developer through the workflow, triggers the QML profiler, crunches through the resulting raw data, presents the performance bottlenecks in a concise report, and suggests improvements.The skill targets 2D Qt Quick applications and supports four profiling modes — rendering, logic, memory, and full. It can also analyze an existing trace file directly, without re-running the application, for example, if the performance trace has been run on the target hardware.
- Union: Spring 2026 Update (2026/05/04 20:10)
  Union: Spring 2026 Update It has been a long time since I wrote anything about Union, the new style engine being developed for KDE. However, that does not mean nothing has happened in that time. Quite the opposite, in fact (spoiler alert); we plan to do a first release of Union with

Plasma 6.7! So let us go over some of the things that happened with Union. CSS as Input Format Image CSS as used by Union for styling buttons and several other controls One of the biggest changes that happened last year is that we switched the default input format from SVG (as used by Plasma) to CSS. Somewhere during spring last year I realised that, while using Plasma's SVG served us well to get the initial data model sorted out, it was now holding things back. Additionally, and maybe even more importantly, it was not something that I felt comfortable with shipping and recommend people build styles with. Plasma's SVG styling, at first glance, looks pretty close to what we would want as an input format: something that would allow designers to easily create and modify styles. The feedback from designers who worked with Plasma's SVG styling was the opposite; getting everything right in the SVG for things to work correctly is a lot harder than it seems, with a number of quirks specific to Plasma that need to be considered, as well as limitations with regards to what features are supported. Additionally, there are several technical issues that made me uncomfortable relying too much on SVGs as an input format. So we wanted to switch to CSS, which has always been the input format I wanted to use. It is well known, has a fairly reasonable syntax and is already being used for the task we want to use it for. However, there is not really a good library available that parses CSS. The best I could find is the Rust cssparser crate from the Servo project. Unfortunately, this is Rust and Union is a C++ project. Additionally, it is slightly too low level for Union to use directly, it is more of a parser-building toolkit than a real parser. So I created the cxx-rust-cssparser library that makes use of the cssparser crate and provides a C++ interface to parse CSS files. Then I hooked that up as a new input format for Union to use and we could start styling things with CSS. Breeze in CSS So we set out to build yet-another implementation of Breeze, this time using Union's CSS. We needed an initial default style for Union, and while there's quite some work happening on a design system and a new style, we wanted a known baseline to work against, similar to what we did with the Plasma SVG input in the first place. This does not mean Breeze will be the only style supported by Union, but rather it limits the scope of the initial work to what is needed for Breeze to work. Image A comparison of four different implementations of Breeze-styled buttons. As it turns out, because we have multiple implementations of the same thing, there are slight differences between those implementations. Some of them have been intentional, such as qqc2-breeze-style using a different spin box style from the Breeze widgets style; others less so. To make it easy to compare what Union's implementation looked like compared to the other styles, I created a simple application that displays pages with controls side-by-side, each page using a different implementation of Breeze. Union's implementation of Breeze is not a one-to-one copy of the Qt Widgets implementation as exposed by qqc2-desktop-style, though it comes quite close. However, in certain cases, we intentionally deviate, either because the different implementations disagree on how to do something, or because Union allows things that we could not do before. Image Comparing different implementations of spin boxes; Union is on the far left. For example, consider the spin boxes above. We have long wanted to use a style where the "increase" and "decrease" buttons are bigger and to the left and right of the value. Both qqc2-breeze-style and Plasma already use this style. However, the QtWidgets style uses a version with small up and down arrows stacked on the right, because a lot of widgets applications expect a spinbox to be fairly small. For Union, we actually implement both: If the spinbox has enough space, we use the preferred style with buttons left and right, but if the spinbox is constrained somehow we switch to the style with smaller buttons stacked to the right. This gives us the best of both worlds, where we can use the preferred style but have a decent fallback for cases where that does not fit. Union's First Release?! Image System Settings' Keyboard page as styled by Union The work on Union's Breeze implementation has progressed to the point where it is very hard to distinguish whether or not you are running the Union version. We have also tested with a bunch of applications and made sure that any differences were fixed. So we are at a stage where we need to get Union into the hands of more people, both to get extra people testing whether there are any major issues, but also to have interested people creating new styles. This means that with the upcoming Plasma

6.7 release, we plan to include Union. Discussion is currently ongoing whether we will enable it by default, but even if not there will be a way to try it out. Looking to the Future I have so far mostly talked about the Qt Quick side of things, but what about Qt Widgets? We so far have focused on the Qt Quick output to have something that we could ship in a good working state. With that work nearing completetion, we can look forward to what to do next. We already have a prototype implementation of a Qt Widgets output for Union. One of the next goals will be to flesh this out further into a complete QStyle implementation that is as usable as the Qt Quick output. Image KDE Plasma using Union for the configuration window. Panel and applet styling is one thing we will add in the future. Another item we will be looking at is to use Union's CSS input for styling Plasma, making it possible for Union styles to provide styling for things like panels and widget popups. Note that we will not drop support for SVG styling in Plasma, but we do hope we can make Union's CSS-based styling the primary way to style Plasma in the future. We also want to expand the things Union can do, so that designers are not limited in what kind of styles they can create using Union. This includes things like expanding the support for specific CSS features, adding support for more rendering options and reconsidering how we handle colours. In Closing As you can see, there is still a lot of work to be done for Union to unionize all of KDE's styling. For that, we would love to hear your experience with it! If you test out Union, please report every problem you find with any existing Qt Quick-based application. See the README for instructions on how to test. You can report issues here. If you are a style designer interested in creating new styles, we are working on documentation that explains how to do so, in the meantime, feel free to experinent with the CSS files used by the Breeze style. If you are interested in working on Union itself, or otherwise would like to discuss something relating to Union, feel free to drop by on Matrix. If you want to get a closer look at what is currently in development, we make heavy use of Gitlab's issues for development tracking, Discuss this post at KDE Discuss. ahiemstra Mon, 05/04/2026 - 22:10

- Introducing the Qt Code Review Skills for Agentic Code Review (2026/05/04 15:12)
  Reviewing, auditing, or sanity-checking code usually means running separate linters, reading through checklists, and manually verifying Qt-specific patterns across dozens of files. The Qt code review skills help developers to automate part of this code review phase. Developers avoid a laborious manual walkthrough of every file, with the AI agent running a deterministic linter followed by six parallel deep-analysis agents and surfacing real issues with mitigations in a few minutes. AI-Powered Code Reviews with Reliable Results
- Goals for GSoC 2026 - Improving Kdenlive Effect Widgets (2026/05/04 11:34)
  Thank you to the KDE community and Jean-Baptiste for selecting my proposal. Congratulations to all other accepted contributors! I'll be working on improving the effect widget system in Kdenlive this summer. As someone who uses Kdenlive daily for my own videos, these are problems I've personally hit, which makes this project feel very personal. Here are my three main goals: Curves WidgetReplace the channel dropdown with a tab-based interface so each color channel (RGB, Red, Green, Blue, Alpha, Luma) has its own independent curve. Currently you need to apply the effect three times to get per-channel control, this fixes that with a single effect instance. Gradient EditorBuild a standalone gradient widget with support for arbitrary draggable color stops, replacing the current hardcoded two-stop system in the effects panel. Speed RampAdd bezier curve handles to the time remapping panel so speed transitions can ease in and out smoothly, with presets like Ease In, Ease Out, Ease In/Out, and Linear. Coding begins May 25. I'll be posting weekly updates here throughout the summer. Looking forward to a productive GSoC!
- Using the WebView module with C++ and Qt Widgets (2026/05/04 10:46)
  For a long time, developers of Qt-based C++ applications have only had one option for embedding web content: Qt WebEngine. And while it offers a large API with many useful features, the module has its downsides, since it can consume a lot of system resources and increase binary

size. For QML users, we've had an alternative in the Qt WebView module, but that API had never been exposed to C++ until now.

- Meet Drawy (2026/05/03 21:30)
  Screenshot This is Drawy, KDE's first infinite whiteboard app written entirely in C++ and Qt. It is inspired by the popular web-based Excalidraw. Its main focus is simplicity, ease of use, and performance. You get all the usual essential features, such as drawing tablet and touchscreen support, basic shapes like rectangles, ellipses, arrows with different arrowheads, lines, etc., text and image support, as well as many other features such as groups, reordering elements, alignment, etc. Why? There's a short story behind this. A few years back, I got myself an XP-Pen Deco Mini 7, which is a drawing tablet for brainstorming and studying digitally. I've been a Linux user for many years, so it was natural that I looked for writing apps made for Linux. I stumbled upon Xournal++, which is an awesome FOSS note-taking app. I should mention that I wanted something for brainstorming and not for note-taking. I used it for a good number of days until I realized that I wanted an infinite canvas, as I hated navigating back and forth between pages whenever I ran out of space. I then discovered Lorien, which is a FOSS whiteboard app that features an infinite canvas. While it did the job, it lacked basic features such as resizing items, and its stroke smoothing felt average. I later tried RNote. It felt like the best app out there and worked flawlessly. The main issue that I found was that its interface required more clicks than necessary to perform certain actions. For example, if you wanted to create an ellipse, you'd need to click on the shapes button -> shape menu -> ellipse. That's three clicks. This may sound like a small issue, but when you want to brainstorm quickly, it gets annoying. It also doesn't allow you to customize the keybindings. I then stumbled upon Excalidraw. It was a dream come true. It was simple, fast, infinite, and had plenty of features. However, it being a web-based app was its biggest downside for me. First, it lags a lot on Firefox. If you are a Firefox user, you will notice that Excalidraw does not offer the same performance as it does on Chromium-based browsers. For some weird reason, this problem is even worse in dark mode. Second, with web-based apps, you lose the ability to save files on your desktop and open them by double-clicking. I'm also a computer science student in my pre-final year, so I was naturally looking for project ideas to work on to strengthen my skills. So I decided that I would make a whiteboard app in C++ because I enjoy working with C++. Qt felt like the natural choice, given that I wanted to work with C++. Development I began developing Drawy in January 2025. I had just learned the basics of Qt and went all in. The first prototype had a finite canvas with basic tools like a pen, some shapes, and an eraser. The code was a mess, but it worked. Later, I learned about low-level design and SOLID principles, so I studied more and refactored the entire codebase around those concepts. This made the code much more maintainable and open to expansion. Development was quite slow, as I only worked on weekends due to college. Many weekends went by without any progress for various reasons, but I never abandoned the project. A fun fact: I'm in a coding club at my college called "Hash Define," where it is a practice for us to teach junior students in the same club. It goes like this: you'll be taught by your seniors, then when you become a senior, you teach your juniors. So I used to teach them concepts related to algorithms, data structures, and object-oriented programming. Since I already had a drawing tablet and my own prototype of Drawy, I used Drawy to teach them. This helped me discover bugs, performance issues, and missing features. I spent my weekends implementing new features that I wanted and fixing bugs and performance issues. Challenge 1: Making it infinite So the first challenge that I faced was making the entire canvas infinite. The first question was, "Where do I even store the items?" My first thought was a list (vector). It is pretty straightforward to store items in a list. When you want to render stuff on the screen, you just traverse through all of them and render. That was a bad idea. Soon, I discovered spatial data structures. There were a lot of them, but I chose the quadtree. It is an awesome data structure that is used heavily in game development and graphics-intensive programs. The main advantage of using a quadtree over a vector is that you can query items in any region in logarithmic time. That sounded really cool to me, so I stuck with it. To make it infinite, I wrote a simple

recursive algorithm that replaces the current node with a bigger node if the query region lies outside the current region. It worked flawlessly. Challenge 2: Performance I made some very poor decisions in the beginning. One of them was rendering content on a QPixmap using the software-based QPainter. I realized this was a problem much later in development, so reverting it was not an option. I decided to keep the current approach and implement algorithms to improve performance. The canvas was infinite, and you could easily add thousands of items to the screen. This was a problem. Adding thousands of items means rendering thousands of items. Since QPainter runs entirely on the CPU, moving around the canvas was unusable. Even with fewer than a hundred items, the lag was unbearable. Almost all of it came from rendering items on every movement of the mouse. The solution was simple: introduce a tile-based cache for the canvas. Items would be rendered once in the cache (which I call the CacheGrid). Now, instead of rendering all the items when moving the canvas, only the cache tiles would be rendered. Since our quadtree allows us to query items by region, we can query items that are visible and render only those on the CacheGrid. Tiles that go outside the viewport are discarded, which keeps memory usage under control. This one change made it go from being barely usable to being smooth as butter. Much later, when I implemented transformations (resize, rotate, and scale), I faced the same problem again. When you transform an item, you cannot reuse its cached version since it has changed. This means that as you transform it, you have to render it again and again. I was devastated. I thought I could not solve this problem at all. I tried switching to a hardware-accelerated backend for QPainter, but it had its own issues, and I did not understand it very well. I then spent a lot of time brainstorming and finally realized that I could use the CacheGrid once again. This time, each item could have its own cache grid. Resizing, rotating, and scaling it would just mean manipulating the cached tiles of that specific item. It involved a lot of math and trial and error, but it worked. Everything was smooth again. Caching was my new love. I implemented similar strategies in more places in the code and made it perform much better, all on the CPU. Performance is not a big issue anymore, at least for now. Becoming a Part of KDE When the alpha version was released, Drawy was still a personal project. It gained 800 stars in the first month of its release. However, there was a lot of stuff I had to implement on my own. For example, a config manager, a keybinding manager, a theme manager, figuring out i18n, distributing packages for different operating systems and architectures, and more. Even with the help of contributors, most of the work had to be handled by me, which would be extremely time-consuming, given that I’m still a student. A few people on Reddit and GitHub suggested that, due to the similar technologies, I should try to make Drawy a part of KDE. KDE has mature infrastructure for distribution and i18n, and KDE frameworks provide convenient functionalities for most of what I needed, so it was an excellent choice. I had also been a KDE user for a long time and had several other projects like Konsave and KDE Control Centre that were built for KDE users, so it felt natural for me to get more involved with KDE. After the incubation of Drawy into KDE, Laurent Montel, who is a senior developer at KDAB and has been with KDE since its very beginning, began contributing to Drawy. He has made a large number of contributions to Drawy, and I’ve learned a lot from him. Drawy would not be what it is right now without his contributions. His contributions include adding the ability to customize keyboard shortcuts, aligning items, implementing a plugin system, adding a color scheme menu, adding infrastructure to configure Drawy, and more. Another contributor who is very much involved with the development is Nikolay Kochulin. He has contributed many features with excellent code quality, such as exporting to images, drag-and-drop functionality, a MIME manager, and support for stylus erasers. His contributions have been very valuable as well. There have also been many other contributors who have helped improve Drawy in different ways. I think it was a really good decision for Drawy to become a part of KDE because I could not have done it alone. Conclusion Developing Drawy has been one of the most instructive and demanding experiences I have taken on. What started as a small personal project gradually evolved into a system that forced me to think beyond just writing code. I had to deal with architecture, performance bottlenecks, maintainability, usability, and long-term scalability,

often all at the same time. Each stage of development exposed gaps in my understanding and pushed me to learn concepts more deeply, whether it was low-level design, rendering optimizations, or managing growing complexity. One of the most important takeaways from this journey is that real-world software development is fundamentally about trade-offs. Early decisions can have long-term consequences, and not every problem has a clean or ideal solution. In many cases, progress comes from working within constraints rather than avoiding them. The transition from a naive implementation to a more structured and optimized system taught me how to reason about systems, not just features. Becoming a part of KDE significantly changed the trajectory of the project. It reduced the burden of handling infrastructure from scratch and allowed me to focus more on the core product. At the same time, it introduced me to experienced developers and higher standards of code quality, both of which had a direct impact on how Drawy evolved. Collaboration played a key role in shaping the project into something more robust than what I could have built alone. Short Note on Problem Solving I must mention that I have spent a lot of my time solving problems on LeetCode, CSES, and HackerRank to improve my problem-solving skills. Even after spending thousands of hours on it, I used to think this was only to “clear interviews,” but developing Drawy has made me realize that this was not the case. Working on your problem-solving skills is extremely important, and these platforms do an excellent job of helping you improve them. I was able to think like an engineer because I had practiced solving these problems before. Being able to write efficient algorithms, focus on maintainability, and make software user-friendly at the same time is challenging and lies at the core of software engineering. Note: No AI was used to write this article. All words are my own.

- Invest in your identity (2026/05/03 20:11)
  I have 30 years of documented history on the web and in my personal recordings. That defines very well who I am, what I do, how I see the world, and how people see me. I worked on that. Sometimes consciously, sometimes as a side effect of my job, my side projects, my community work. Now that AI agents make it easy to use this kind of material, I have a base to anchor them, to build on what I did before and accelerate what I do, still staying me.If you are starting now, you won't have this body of material to anchor your agents. So do spend some time building this corpus of what is genuinely you. Don't let an AI generate what you are. Write yourself, publish, think through your thoughts, give presentations. Small things are fine. They will accumulate over time.Of course, tools will shape part of your identity. I used to do my presentations with xfig, printed on overhead projector slides. This was painful, but it shaped quite a bit how I worked and how the result looked. So it is part of my identity. The technical constraints did influence how I spoke, how I presented. It also shaped what I presented, because there was a bias toward what I could show with the tools available to me.This won't be different with AI. It will shape who you are. But be aware, and make sure that there is a signal from the human in there. It's ok if it's imperfect, if it's a bit weird. It's ok if it's different. But make sure it's yours.Shape that signal. That's you. That's your identity.
- KDE email, part 3: don’t filter your email (2026/05/03 17:03)
  This is part 3 in my series about email management, with the prior one being about using email client apps. This one is about trying to use email filtering to handle email overload. You’re getting too much email It’s a flood — no, a deluge! Hundreds of messages a day. Overwhelming. Demoralizing. Soul-crushing. The thought of even looking at your email provokes anxiety. What to do? Email filtering to the rescue! Use sieve (KMail even includes an app for it!) to implement a bevy of server-side filtering rules that send emails to different folders. So neat and tidy. So clean. So organized. So much better… not! Filtering doesn’t work You started with the problem of “I get too much email to comfortably handle”. With filtering, you’ve split up the “too much email” into multiple folders, but all those folders put together are still impossible to comfortably handle. You may have told yourself that this system helps you prioritize, because the most important emails go to your inbox. But it’s not true; an

email's importance can have much more to do with its content than the characteristics you're probably using for filtering (sender, mailing list ID, subject line, etc). For example: You commented on a bug report, and then someone else replied to your comment with a question. The email notifying you about their reply got filtered into oblivion, so you missed it, and now that person thinks you're rudely ignoring them, or negligent, or incompetent. That's damage both to your reputation, and to KDE's. This is what leads people to whine "KDE doesn't care!" on social media. Also, the properties you filter against change over time, which means mail filtering requires maintenance to keep the important emails in your inbox — maintenance that you'll eventually tire of doing and neglect. Which means some important emails will still be shunted away to folders you aren't checking regularly. Which means you're still missing them. Which means filtering hasn't solved the problem of missing emails and being perceived as unreliable or rude. I get it. Filtering is tempting. But it's just covering up the actual problem. There are only three real solutions to "too much email": 1. Spend more time processing emails For a busy professional like you, email is a task list that other people can add items too. This is terrible, but it's also a professional obligation, so you need to block out time to handle those tasks somehow. Yet spending tons of time on it will burn you out! So minimize this to only what's absolutely necessary to avoid your inbox becoming more full over time. Make the "number of emails in the inbox" trend-line negative. Which means you need to… 2. Get fewer emails Every minute you put into reducing emails will pay you back 100x over the next few years. The project you're regularly working on or monitoring via a website? Turn off email notifications; you'll see stuff on the website. That mailing list for a project you haven't had any involvement with in years? Unsubscribe. Merge requests for a project you're only tangentially interested in? Un-watch in your notification preferences. Notifications about things happening in real-time? Switch to a daily digest in your email preferences, or unsubscribe and set aside a time to check that thing manually. Marketing emails for literally everything? Unsubscribe. News? Unsubscribe unsubscribe unsubscribe! You'll learn about anything important in another way. Emails about bills and payments you have to make? Put them on auto-pay, then delete the "payment submitted" emails without even looking at them. And so on. In the "email as task list" model, you have to reduce the number of people, groups, and companies who can assign you email-tasks, or you'll go mad. Do it, do it now! C'mon, kill those emails! 3. Increase speed of processing emails A key part of this is using an email client, which I wrote about earlier. Learn your tools! Use keyboard shortcuts. Aggressively delete and archive emails after you handle them. I like automatic color tagging, which I wrote about back in 2024. There are lots of techniques to process emails faster, and I'll write about some of them later. But focus on solving the problem, rather than hiding it. The important part is to see email as a job skill you can commit to getting better at, just like programming, debugging, or source management using git. Don't accept that you suck at email, give up, and hide the problem under the rug. Get better! Filtering is a tool that holds you back and prevents you from learning stronger email management skills. Ditch the filtering habit. It'll be hard at first, but you can push through that and solve the real problems of too much email, un-optimized workflows, and fear of managing email due to lack of the first two. You can do it! I believe in you!

- Gestures in Graz, and beyond (2026/05/03 09:55)
  KDE's Mega Sprint 2026 in Graz brought a group of about 20 KDE contributors together in early April, to discuss technical challenges, make decisions, and get stuff done. With travel support from KDE e.V. (thanks to your donations), I was able to join the group there. Other blogs have reported on their experiences of what happened in Graz: Kieryn Darkwater, Albert Astals Cid, Kristen McWilliam, Volker Krause, and Raresh Rus. A large variety of topics were on the agenda, people were wearing multiple hats to help each other out. I, on the other hand, went to Graz with one sole purpose: figure out the best way to merge gesture customization. Dude, what's taking you so long Quick recap. Since NLnet approved a grant last year, we've published requirements, designed settings UIs, proposed config file formats, ported my earlier mouse gesture prototype to

KWin, and presented at Akademy 2025. Then I got injured for a while, hit a slump through the fall and winter, and didn't make much progress until more recently. Meanwhile, as an actual Plasma user, you have benefited from exactly none of this prep work. For example, a representative sentiment from the comment section of Brodie Robertson's recent Plasma 6.7 upcoming features video: Yet again it looks like configurable touchpad gestures have slipped.... To be fair, we haven't actually promised any particular Plasma release for this to land in. But I get it. It's been weighing on me probably harder than it's been weighing on you. So I finally emerged from the slump with a set of experimental patches that allowed real-time ("one to one") gestures like Overview to be reassigned to a different gesture via config file options. Quick architectural primer There are several KDE components that work together to handle keyboard shortcuts: KGlobalAccel: This KDE Framework is used by an app or desktop component to register actions and their preferred global shortcuts. KGlobalAccel doesn't actually do much by itself, it just sends requests to a background service (via D-Bus) and tells the app what shortcut actually got assigned for each action. Then it just waits for a signal from the background service that the shortcut was pressed and the action should be performed. KGlobalAccelD: This is the background service! Part of any Plasma desktop. It listens for global shortcut requests from apps, looks for additional shortcuts in *.desktop files, and reads from and writes to ~/.config/kglobalshortcutsrc to manage shortcut assignments. It can check if a given key combination will trigger a registered action. If it does, it tells the corresponding app to perform the action. On Wayland, KGlobalAccelD is embedded into KWin, giving KWin access to functionality that isn't available over D-Bus. KWin: Plasma's compositor which we all know and love. It manages windows and puts their contents on screen, but it also has exclusive access to libinput on Wayland. So any key presses, mouse clicks, touch taps, and pointer movement will go through KWin. Most of these just get relayed to the current app. But some of them are intercepted. For global shortcuts, KWin asks its KGlobalAccelD service to check if it should eat up the shortcut and fire a global shortcut instead of sending it to the currently focused app. System Settings / "Shortcuts" page: An independent app that asks the KGlobalAccelD service for all components with registered shortcut actions. If the user changes a shortcut assignment, it tells the KGlobalAccelD service about it. As a result, KGlobalAccelD notifies the app that its shortcut has changed. XDG Desktop Portal KDE: Only tangentially involved. It implements the Global Shortcuts portal, so that any app can register global shortcuts, even if they aren't using KDE Frameworks. All it does is to translate an app's portal requests to requests for KGlobalAccelD. The consensus is that gestures are much like keyboard shortcuts, so gesture configuration should be implemented in similar ways, using the same components. There are important differences of course. How to deal with those isn't quite trivial. That's where sprint discussions come in. Back to Graz Having a functional prototype is a good start, but it's still a couple of steps removed from getting it merged into Plasma. I arrived with a number of technical questions to hash out the exact plan and get agreement. All the minutious details can be found in the sprint notes wiki page, but the short version is that the implementation plans became very clear. There were three breakthroughs in particular: We don't need to expose gesture details through KGlobalAccel: System Settings can link against KGlobalAccelD directly, so we can put the new trigger class in there without worrying about backwards compatiblility for apps. Natalie, Nate, Kristen and I sat down for a long UX design session, improving on our previous UX proposal together. Nicolas Fella provided wise advice on config syntax, leading to a better trade-off between readability, clean code, and migration concerns. Xaver Hugl deserves a special mention, he was generous with all the other technical questions that I kept pestering him about. Of course I also got involved in a few other discussions, as well as daily dinner outings. It's a good group. But you've read about this on other blogs already, so I won't bore you except with a few more pictures. Proof I was there I bought an amigurumi Konqi One of many food outings Walking to a rustic restaurant on a hill Snackies Konqi at Linuxtage Graz I spent the remainder of April improving my previous patches. The result is a series of MRs that are finally ready for code review by KWin and KF6 maintainers (1, 2, 3, 4, 5). This will allow assigning touchpad

and touchscreen gestures for global actions by editing the kglobalshortcutsrc config file. Schedules were tight, so this didn't make the cut for Plasma 6.7 anymore. Let's see if we can get it into 6.8 though. Based on this work, my work-in-progress changes to support mouse gestures stroke gestures line shape gestures have been improved as well (1, 2). Autotests and more code polish are needed before this can go into code review, but it's now quite functional: path simplification, shape recognition, action configuration, and a neat little visualization effect are all in place. Other recent input highlights Plasma Keyboard was also heavily discussed in Graz, spearheaded by Kristen who put some serious polishing work into it. Kristen introduced a long-press diacritic selection pop-up for Plasma 6.7, a killer feature in my opinion. Meanwhile, on a different continent, Devin Lin has been hard at work to prepare an architectural overhaul of Plasma Keyboard, which would facilitate much-requested improvements to the on-screen keyboard experience (this won't make it into 6.7 though). Quinten Kock's changes to make touchpad pinch zoom gestures work in Okular are now available with the recent KDE Gear 26.04 release. This bug was pointed out in the discussion on our Input Goal proposal, all the way back in 2024, so it's great to see it fixed. If you're a developer, have a look at the commit and use QNativeGesture events to support touchpad pinch. Shout-out also to Taufeeque Sifat, who's been improving Okular's text selection behavior including triple-click line selection. Kai Uwe Broulik modernized the code for mouse and touchpad settings pages. This is not a user-visible change, but future changes to these settings become easier thanks to Kai Uwe's work. I'm excited because it paves the way to a unified Mouse & Touchpad settings page in future releases. Alexander Wilms tweaked System Settings to hide Mouse, Touchpad, and Game Controller settings pages when no such devices are connected. Also notable is work on a custom pointer acceleration curve editor, which is not ready to ship yet but hopefully can get there eventually. Joshua Goins added an option for the stylus cursor to stay in sync with the mouse pointer. David Redondo fixed the stylus button assignment lines looking weird in the Drawing Tablet page in System Settings. Vlad Zahorodnii added support for libinput plugins to KWin, allowing technical users to work around otherwise unfixable input hardware quirks. Xuetian Weng and Nicolas Fella improved the behavior of key repeats for input methods and screen readers. David Edmundson fixed an issue with pointer positions for mirrored touchscreen displays. Thanks to everyone who has worked on KDE's Input Goal in the past! At this year's Akademy in Graz, we should see a new set of goals elected, so we're in the final stretch for this one. Onward to Germany I wrote this blog post while riding a long-distance train, on the way to another gesture customization mini-sprint with Natalie. The plan is to focus on implementing the new settings pages that will let regular users reassign (or unassign) gestures. Thanks again to KDE e.V. for travel support for another week of in-person collaboration. Let's see how much stuff we can get squared away - it's not just settings forms, but more infrastructure on the KWin and KGlobalAccelD side is also still needed to make everything fit together. There will be another blog post to follow up on this later. Until then!

- Dolphin 26.04 release (2026/05/03 00:00)
  I want to highlight a few changes that came to Dolphin 26.04 and add some nuance to the release announcement. The the KDE Gear 26.04 release announcement mentions: In version 26.04, Dolphin lets you add keyboard shortcuts to nearly any option in any menu, plugin or extension. This refers to this MR: Add keyboard shortcut support for service menu actions. This change made it possible to add shortcuts to the beloved service menus and closed a highly-requested feature. In other words, this allows you to add shortcuts to custom actions that can be run on currently-selected files or folders in Dolphin. This was implemented by a new contributor Albert Mkhitaryan, who also did the necessary library work with my assistance. Kudos to Albert. Most of Dolphin's UI could already benefit from shortcuts with some caveats and the new feature does not apply to context menu plugins, such as Ark's "Extract here" or Dolphin's "Set Folder Icon". Additionally, context menus options are reloaded as they are modified thanks to Pan Zhang. xi ota added an option to pick the tab width. There was an important fix for split view users by Māris

Nartišs. I made a small improvement to the layout of the Settings dialog. Rafał Lichwała improved dragging and dropping from the Places panel. Of course a few bugfixes (not exhaustive list): 514209 preventted flickrering when holding F5 key, by Ritchie Frodomar, 510829 improved the rendering of gif/animated images in hidpi context, by me. 453262 refreshing the image preview in the information panel after a file is renamed, by me. And we reverted a dreaded change to the Create New Folder shortcut. Another thing worth mentioning is the memory leak detection effort started by Nicolas Fella, as this allowed me to find a couple of uncommon mem leaks. The work is not complete as KIO CI tests don't pass under LSAN scrutiny yet. I have been working on it and only a couple of hard cases remain to be solved - help welcome. Next will be activating LSAN for Dolphin tests and CI for further stability but that is already fruitfull. Having stopped my personal blog, I am now using kde-blogs. That's all, folks.

- Goals for GSOC 2026 and Mankala Engine (2026/05/02 16:01)
  Firstly, thank you to the entire community and mentors for selecting my proposal for GSOC. Congratulations to all others Goals for GSOCStarting with the goals for ManakalaNextGen, the GUI of Mankala Engine, the main goal is to implement a tournament system for the game. I plan to start with improving the user registrations by giving users the option to create an XMPP account directly from within the game. We can have a minimum of 3 servers, which the game can support. Based on this, we can also have possible player icons and in-game names for the players, which would be displayed in matches. Now, coming to the important part for tournaments, my aim is to create a new page for the tournament host and the participants. In that host specifies the game variant like Bohnenspiel, Oware, etc., and also the amount of time each game should run. The participants need to join the game using the room code given by the host. At last, we implement the logic for player elimination and create a leaderboard ranking the players based on their position. Our main concern?Yes, the game data might get lost. We need to come up with suitable solutions to export it and make sure that even if a player leaves the game, the game data is not lost and isn't declared invalid. So, to summarize this, here are the main goals: Add XMPP server registration option for players and update the registration page. Create tournaments and logic for player elimination and ranking. Make other necessary changes based on feedback. Thanks for reading. Looking forward to a productive summer
- This month in KDE Linux: April 2026 (2026/05/02 13:27)
  Welcome to another edition of “This month in KDE Linux”! Infrastructure remained a major focus this month, with multiple outages and bugs in Arch’s package archive leading to Harald Sitter creating a local mirror for KDE Linux. This substantially increased build delivery reliability. Harald also worked on improving the speed of delta updates. This is experimental and in-progress, so you have to opt in; See the bottom of https://community.kde.org/KDE_Linux/Delta#Status Beyond that, a number of features are under development but did not quite complete yet, so expect to hear about them next month. This month, Hadi introduced a terminal handler to prompt you to add execute permissions to scripts lacking it when you try to run them: Hadi also moved our console handling to the newer userspace Kmscon software, which we’re using in place of the built-in console from the Linux kernel. Text looks way better now! Thomas Duckworth implemented screen reader support for the installer. Jonas Harer and Daniele Me made the default zsh config even better. It really is a joy to use now! Aidan turned on IPv6 privacy addressing by default, improving privacy a bit when using IPv6 connections. I made KDE’s ksshaskpass dialog be the thing that prompts you for the password to unlock your encrypted ssh keys, which also allows you to have it save them in the system’s password storage system if you’d like. I also simplified the process of setting up an ssh agent to automatically add your keys, and documented how to flip the final switch to turn it all on. I also documented how to persistently change kernel parameters, in case you need some extra ones (for example, turning on the experimental Xe

driver for your newer Intel GPU). Finally, I flipped the switch to have KDE Linux use the new Union theming system by default for QML apps. If the results in non-Flatpak QML apps like Discover, System Settings, Info Center, and Emoji Picker look no different… that's perfect! That's all for April, folks! I'll see everyone for the May report, or ideally, sooner. Because, as you can see, while KDE Linux is being developed by multiple people (good for project health), the number of changes is a bit low (bad for project velocity). There's plenty to do, so if you're a fan of the project, please help out: If you're an adventurous and technical person, install KDE Linux and report issues. If you're good at writing, KDE Linux's documentation can always use improvement. Submit merge requests here. KDE Linux leans heavily on Flatpak, so fixing packaging or code issues in Flatpak-packaged apps is very helpful. You can even help us build the OS itself! The Beta milestone is currently 73% complete, and there's plenty to do. The Incus-based Kapsule system is integral to our "expansion by experts" story. If you're a container expert or low-level OS nerd, working on the child tasks here is hugely impactful And if you're already using KDE Linux, let us know how your experience has been! Is it good? What can we do better?

- This Week in Plasma: Background Apps and Zoom Up-Scaling (2026/05/02 00:00)
  Welcome to a new issue of This Week in Plasma! This week Plasma 6.7 entered its "soft feature freeze" where we stop merging newly-written features and focus on finishing up and merging the ones that were already in flight. As such, some nice new features that have been in development for quite some time were merged this week! In addition, Plasma got a number of nice quality-of-life UI improvements and some accessibility fixes, among other changes. A good haul this week: Notable new features Plasma 6.7 Implemented support for the "Background apps" portal. This allows apps (especially newer GNOME apps) that use this portal to put themselves in the background and appear as icons in the System Tray, alongside the similar icons for other apps that use the existing System Tray icon functionality. (David Redondo, plasma-workspace MR #5703) Implemented an up-scaling feature for screen content when using KWin's Zoom effect. The filter does its best to sharpen and upscale content, resulting in a smoother and less blocky appearance, especially at relatively high zoom levels. If this effect isn't for you, you can turn it off. (Ritchie Frodomar, KDE Bugzilla #509770) Medium magnification High magnification Massive magnification The Printers widget is now badged with the number of active and queued print jobs. (Mike Noe, print-manager MR #323) Notable UI improvements Plasma 6.7 XWayland-using software that asks to be able to send synthetic keyboard and mouse events (such as xdotool, which it turns out a bunch of apps invoke) is now identified by name so you know what's asking. In addition, you can see a list of apps you've given this permission to, and revoke it later. (David Redondo, kwin MR #9123) Implemented some KDE styling to the generic MessageDialog component from Qt, which resolves the issue of these dialogs looking really ugly and out of place in various pieces of software, including the Sticky Note widget's deletion confirmation dialog. (Tobias Fella, KDE Bugzilla #499562) After Before Improved how Discover handles being launched with no internet connection. (Tobias Fella, KDE Bugzilla #511002) Improved how Discover communicates that a firmware update has been queued for installation after the next restart. (Tobias Fella, KDE Bugzilla #422498) Removed the "double back button" effect visible in the Networks widget when showing a network's QR code. (Tobias Fella, plasma-nm MR #541) Made the automatic screen brightness feature take into account more data points, hopefully making it more responsive to your desires and less swingy when in an environment where the background lighting is changing a lot. (Prajna Sariputra, kwin MR #9145) Made the buttons at the top of the Widget Explorer sidebar respect Fitts' Law, allowing you to activate them by jamming the pointer against the adjacent screen edge and clicking. (Tobias Fella, plasma-desktop MR #3511 and libplasma MR #1479) Streamlined the presentation of the notification about your KDE-Connect-connected phone being low on battery power. (Kai Uwe Broulik, powerdevil MR #619) You now have more than 25 seconds to pick a color once you've invoked this from the Color Picker widget. Now you can

take as long as you like. (Kai Uwe Broulik, kdeplasma-addons MR #1013) Frameworks 6.26 Improved the appearance of the cross-fade transition when moving between pages in many Kirigami-based apps. (HeCheng Yu, kirigami MR #2079) Notable bug fixes Plasma 6.6.5 Fixed an issue that made Plasma Login Manager fail to launch properly on certain devices with certain graphics hardware — in particular Apple silicon laptops. (Matthias Kurz, plasma-login-manager MR #130) Fixed an issue that made touches on a touchscreen stop applying to the correct part of the screen when the touchscreen was mirrored to another screen with different geometry. (David Edmundson, KDE Bugzilla #514688) Made it possible to select a sound theme using the keyboard. (Nicolas Fella, KDE Bugzilla #519194) Fixed a visual glitch on System Settings' Drawing Tablet page that made the lines indicating mappings for stylus buttons fly off the top-left corner of the window with certain tablets. (David Redondo, KDE Bugzilla #519600) Plasma 6.7 Fixed a case where KWin could crash when activating an item on a hidden panel while using the in-development Vulkan rendering backend. (Vlad Zahorodnii, KDE Bugzilla #518721) Fixed two cases of controls on System Settings' Quick Settings page not being read by the Orca screen reader. (Nicolas Fella, KDE Bugzilla #519433) Disabling KRunner plugins globally now turns them off in the Kicker Application Menu widget, too. (Christoph Wolk, KDE Bugzilla #501200) Notable in performance & technical Plasma 6.6.5 Fixed some performance issues experienced on a variety of NVIDIA GPUs that were introduced by version 595 of the proprietary NVIDIA driver. (Xaver Hugl, KDE Bugzilla #517987) Plasma 6.7 Implemented support for renaming or relocating the new cross-desktop standard "Projects" folder that's started to appear in people's home folders. (Jakob Dev, plasma-desktop MR #3657) Implemented version 2 of the "Input Capture" portal. (David Redondo, xdg-desktop-portal-kde MR #493) How you can help KDE has become important in the world, and your time and contributions have helped us get there. As we grow, we need your support to keep KDE sustainable. Would you like to help put together this weekly report? Introduce yourself in the Matrix room and join the team! Beyond that, you can help KDE by directly getting involved in any other projects. Donating time is actually more impactful than donating money. Each contributor makes a huge difference in KDE — you are not a number or a cog in a machine! You don't have to be a programmer, either; many other opportunities exist. You can also help out by making a donation! This helps cover operational costs, salaries, travel expenses for contributors, and in general just keeps KDE bringing Free Software to the world. To get a new Plasma feature or a bug fix mentioned here Push a commit to the relevant merge request on invent.kde.org.

- Kirigami forms and configurations (2026/04/30 15:54)
  Recently a new submodule has landed in Kirigami: "Forms". Until this point, Kirigami had only offered the classic "FormLayout" component. which is used for configuration pages throughoug systemsettings, Plasma, and some apps. It's the classical form used in desktop toolkits for decades: This is a fairly clean layout which however is starting to slowly become outdated, as modern toolkits are starting to use a different layout nowdays, based on "cards" Unfortunately FormLayout very bound to the classic layout, and it wasn't really possible to adapt it to the new look in a compatible way which didn't break existing applications in unexpected ways. This is also the reason a new approach was done provided by kirigami addons: "FormCard", which is used by a lot of applications; for instance here in NeoChat: We wanted to have this new style of forms in the base Kirigami API, so after a review of the existing FormCard, we decided to make several changes, for two main reasons: First, FormCard is bound to the card style of form as much as FormLayout was bound to the classic flat style. Also, it tried to provide ready-made components for every kind of control; so it had its own TextField, its own RadioButton and so on — effectively becoming its own separate toolkit. So we opted instead to go down the route of having a more generic API, so the Forms module includes containers that define a semantic structure of a form, which contains all the normal controls — such as textfields, checkboxes and radiobuttons. This is a code example of the new API: import QtQuick.Controls as QQC import org.kde.kirigami as Kirigami Kirigami.Form { Kirigami.FormGroup { title: "Global Settings" Kirigami.FormEntry {

contentItem: QQC.CheckBox { text: "Show Sidebar" } } Kirigami.FormEntry { contentItem: QQC.CheckBox { text: "Auto Save" } } } Kirigami.FormGroup { title: "Theme Options" Kirigami.FormEntry { title: "Colors:" contentItem: QQC.CheckBox { text: "Dark Theme" } } Kirigami.FormSeparator {} Kirigami.FormEntry { contentItem: QQC.CheckBox { text: "High Contrast" } } ... } } which will look like this: Or, in mobile mode: Semantically, a Form will contain one or more FormGroup objects, each of which will contain one or more FormEntry objects. Then a FormEntry will contain the control which represents the configuration of the particular thing. It can be a single control (like a button or a checkbox) or it can be any layout with completely custom contents. I already ported 4 modules of systemsettings to the new system: the landing page, the “workspace options” kcm, the mouse settings and the touchpad settings. But wait… this page looks exactly the same as before; why? A key was to do an API that was as much as separated from any appearance as possible, as we don’t know how UI design trends will evolve in the future. But this also allows us another thing: to have two separate implementations: the new one “card based” and a legacy one which looks exactly like the current FormLayout components. This is used only in systemsettings, so we can port all the kcms without introducing glaring visual inconsistencies, and when we are done, flick the switch and convert the look of everything all in one go. Since most of KDE’s QML applications already use the existing card-style kirigamiaddons FormCard components, the new look will be used there. And then in the future, when trends change again, we can re-style all the settings pages in one go. A call to action We ideally want the whole set of systemsettings kcms to be ported as soon as possible to the new system, so we can have soon a nice visual overhaul in the whole systemsettings. In order for this to happen, we need the help of everyone, so… patches welcome As an example, this is the merge request that ported the first four kcms. When porting, it’s also possible to see how the kcm will look with the new system as well, to make sure it works well for when we flick the switch. If we run in a terminal: KDE_KIRIGAMI_FORMS_STYLE=cards systemsettings We get systemsettings using the new style for pages already ported: Porting from FormLayout to the new Form/FormGroup/FormEntry system should be really straightforward; it should be possible to make good progress in little time. With your help, soon KDE’s settings will benefit from greater consistency, a more modern style, and easier adaptation to future designs.

- Qt Contributors Summit 2026: Oslo in October! (2026/04/30 08:37)
  Hello Qt,
- KDE & Google Summer of Code 2026 (2026/04/30 00:00)
  Google Summer of Code (GSoC) is a training/mentorship program that allows new contributors to open source to work on projects for between 175 to 350 hours under the guidance of experienced mentors. KDE will mentor twelve projects in this year's Google Summer of Code. Marknote Marknote is a rich text notebooks organizer. Prayag Jain aims to introduce a robust and high performance block editor and a proper markdown parser to Marknote under the guidance of Carl Schwan and Mathis Brüchert. digiKam digiKam is an advanced open-source digital photo management application which provides a comprehensive set of tools for importing, managing, editing, and sharing photos and raw files. Srirupa Datta, who already successfully completed a GSoC in 2023 working on Krita, will this year work on digiKam to interface the database search engine to an AI-based LLM. This project is mentored by Gilles Caulier, Michael Miller and Maik Qualmann. Drawy Drawy is a recent addition to KDE applications! It is an infinite whiteboard tool useful for brainstorming. Abdelhadi Wael will improve the text items by implementing rich text handling and other improvements. This project is mentored by Laurent Montel. Kdenlive Kdenlive is KDE's video editor. Yash Bavadiya will work on improving the curves, gradients and time remapping effects. This project is mentored by Jean-Baptiste Mardelle. Kirigami Shubham Shinde, a successful participant in last year's GSoC working on Merkuro, will work this year again under the guidance of Carl Schwan to improve the

Kirigami framework and the developer experience. Okular Okular is KDE's universal document viewer. Ojas Maheshwari will work under the guidance of Albert Astals Cid to implement font subsetting when saving PDF files in Poppler (the PDF rendering library used by Okular). Lokalize Lokalize is the localization tool for KDE software and other free and open source software. Navya Sai Sadu will improve Lokalize by redesigning the translation memory tab to help with the translators' experience. This project is mentored by Finley Watson. KStars KStars is the KDE astronomy software providing an accurate graphical simulation of the night sky, from any location on Earth, at any date and time. Pavan Kumar S G will add a new AI powered guiding assistant for Ekos under the guidance of Jasem Mutlaq. KeepSecret KeepSecret is a password manager for viewing, editing, creating, or deleting passwords. Roshani Kumari will work on improving the user experience and adding new features such as import/export of passwords, adding a built-in password generator, and much more. This work is mentored by Marco Martin. Mentorship Portal Ansh Singhal will work on creating a new join.kde.org website which aims to improve the onboarding experience by centralizing the different entry points the KDE community has. This project is mentored by Anish Tak, who was a successful mentee last year on the same topic! Plasma Mobile Tushar Gupta will rework some networking modules (KCM) to make them more mobile friendly. This work will be mentored by Carl Schwan and Devin Lin. Mankala Mankala is a two-player board game containing multiple variants. Sayandeep Dutta will add a new interface to create tournaments for the Mankala game. This project is mentored by Benson Muite and Srisharan V. S. who completed a successful GSoC contributor on the same project last year.

- Introducing Qt Agentic Development Skills (2026/04/28 06:04)
  Today, we are releasing the first set of skills for agentic Qt development, designed to multiply your productivity when writing, documenting, and reviewing Qt code. If you want to know more about Qt's vision for agentic development and what agentic development for Qt is, then do check out the related article here: Software Insights
- What’s new for Fedora Atomic Desktops in Fedora 44 (2026/04/27 22:00)
  Fedora 44 has been released! □ So let’s see what is included in this new release for the Fedora Atomic Desktops variants (Silverblue, Kinoite, Sway Atomic, Budgie Atomic and COSMIC Atomic). Note: You can also read this post on the Fedora Magazine. Changes for all Atomic Desktops Issue tracker moved to the new Fedora forge We have moved the cross-variants issue tracker to the new Fedora forge. This is the best place to file issues that impacts all variants or to coordinate work between all of them. If you have issues specific to a given desktop environment then we usually prefer to track them in each respective SIG trackers. They are listed on the README for the atomic-desktops organization. Unified documentation, hosted on the new forge The unified documentation for all Atomic Desktops is finally live! Unfortunately the translations have not been migrated so we will need help to re-translate everything again, once the translation setup is ready with the new forge. It should be mostly copy/paste from the previous docs and this time we will only have to translate the docs once and not for every (new) variant. See the tracking issue atomic-desktops#10. Removal of FUSE version 2 libraries FUSE version 2 has been deprecated and unmaintained for a while so we have removed it from the images. In practice, this means two things: If you are using AppImages, some of them may not work anymore. If you are using legacy backends with Plasma Vault on Kinoite, you need to migrate your data. See the Fedora Change and the tracking issue atomic-desktops#50. The implications are detailed below. AppImages and the FUSE 2 libraries Some AppImages are still using an old AppImage runtime that relies on FUSE 2 libraries being available on the host. See the discussion thread for examples on how to check the runtime of an AppImage. If some of your AppImages do not work on Fedora Atomic Desktops 44, we recommend: Looking for a Flatpak for the application and giving it another try. Consider helping upstream package their application as a Flatpak. Reporting the issue upstream so that they are aware that they

should use a newer runtime. Consider helping upstream with this as well. EncFS or CryFS backends for Plasma Vaults are removed KDE upstream no longer recommend using the EncFS nor CryFS backends for Plasma Vaults, notably because they rely on the FUSE 2 libraries. If you are using one of those backends, you should migrate your data to a new Vault using the only maintained backend (gocryptfs). Ideally this should occur before the update to Fedora 44. If you have already updated to Fedora 44 and need access to your data, you can layer the needed packages (cryfs or fuse-encfs) using rpm-ostree install <package>, then migrate your data and finally reset the layers with rpm-ostree reset. Dropping compatibility for pkla polkit rules Support for the legacy pkla polkit rules format has been removed. It is unlikely that you were relying on support for those rules as most of the ecosystem has moved on to the new Javascript based format. See the Fedora Change and the tracking issue atomic-desktops#102. What's new in Silverblue GNOME 50 Fedora Silverblue comes with the latest GNOME 50 release. For more details about the changes that alongside GNOME 50, see What's new in Fedora Workstation 44 on the Fedora Magazine. What's new in Kinoite KDE Plasma 6.6 Fedora Kinoite ships with Plasma 6.6, Frameworks 6.24 and Gear 25.12. See also What's new in Fedora KDE Plasma Desktop 44 on the Fedora Magazine. KDE Plasma Login Manager replaces SDDM The brand new Plasma Login Manager replaces SDDM to provide a more integrated experience with systemd and the KDE Plasma session. See the Fedora Change. Unified out of the box experience with KDE Plasma Setup (OEM installation) Thanks to the new Plasma Setup, it is now possible to install the system with Anaconda with minimal configuration and then complete the installation on the first boot by creating a new user and selecting the timezone. This is great when you want to install Fedora Kinoite on a computer and don't want to setup a user in advance. See the Fedora Change. What's new in Sway Atomic Nothing specific for this release. What's new in Budgie Atomic Fedora Budgie Atomic comes with the latest 10.10.2 Budgie release. This release brings Wayland support to Budgie Atomic. See the 10.10 release announcement for more details. What's new in COSMIC Atomic Fedora COSMIC Atomic comes with the latest 1.0.8 release of the COSMIC desktop. This is now considered stable. Universal Blue, Bluefin, Bazzite and Aurora Our friends in the Universal Blue project (Bazzite, Bluefin, Aurora) have prepared the update to Fedora 44. Look for upcoming announcements in their Discourse. As always, I heavily recommend checking them out, especially if you feel like some things are missing from the Fedora Atomic Desktops and you depend on them (NVIDIA drivers, extra media codec, out of tree kernel drivers, etc.). What's next Helping us with a few nasty bugs If you are interested in contributing to Fedora Atomic Desktops, here are some bugs that we will have to fix in the short term. We would greatly appreciate help with: Fixing root mount options (atomic-desktops#72): This is a long standing and mostly invisible bug that impacts performance. Moving away from nss-altfiles (atomic-desktops#108): This is another long standing source of issues that new users regularly face. Sealed Fedora Atomic Desktop bootable container images Sealed images are now ready for testing! See the other article for all the details. Roadmap to Bootable Containers A lot of work is happening to make the transition to Bootable Containers as smooth as possible for our existing users. You can look at the roadmap for this transition at atomic-desktops#26. One of the tasks is to move away from our unmaintained installation ISO building scripts to the new image-builder tooling. This is planned for Fedora 45 for the ostree variants and support for Bootable Container will follow right after. Another task is to start building the Fedora Atomic Desktops Bootable Container images using the Fedora Konflux instance. Where to reach us We are looking for contributors to help us make the Fedora Atomic Desktops the best experience for Fedora users. Atomic Desktops SIG: Organization on Fedora's Forge, #atomic-desktops:fedoraproject.org Silverblue: Workstation Working Group, #silverblue:fedoraproject.org Kinoite: KDE SIG, #kinoite:fedoraproject.org Sway Atomic: Sway SIG, #sway:fedoraproject.org Budgie Atomic: Budgie SIG, #budgie:fedoraproject.org COSMIC Atomic: COSMIC SIG, #cosmic:fedoraproject.org

- Sealed Fedora Atomic Desktop bootable container images (2026/04/27 22:00)

I’m happy to announce that we have sealed bootable container images ready for testing for the Fedora Atomic Desktops! Note: You can also read this post on the Fedora Magazine. What are sealed bootable container images? Sealed bootable container images include all the components needed to create a fully verified boot chain, from the firmware to the operating system composefs image. This relies on Secure Boot and thus only supports system booting with UEFI on x86_64 & aarch64. The components are: systemd-boot as bootloader, a Unified Kernel Image (UKI) which includes the Linux kernel, an initrd and the kernel command line, a composefs repository with fs-verity enabled. This is managed by bootc. Both systemd-boot and the UKI are signed for Secure Boot. The images are test images so the components are not signed with the official keys from Fedora. The main direct benefit that we will get from this support is that we will be able to enable passwordless disk unlocking using the TPM in a way that will be reasonably secure by default. How do I test those images? See the instructions at github.com/travier/fedora-atomic-desktops-sealed on how to give the pre-built container and disk images a try and how to build your own. We welcome testing and feedback! Please see the list of known issues and report new issue at github.com/travier/fedora-atomic-desktops-sealed. We’ll redirect them as needed to the right upstream projects. Beware, those are testing images. The root account does not have a password set and sshd is enabled, by default, to make debugging easier. The UKI and systemd-boot are signed for Secure Boot but, since those are test images, they are not signed with the official keys from Fedora. Don’t use those images in production. Where can I get more details about how this work? If you want to know more about how sealed images work (i.e. how we make bootable containers, UKI and composefs work together to create a verified boot chain), see the following presentations and documentation: “Signed, Sealed, and Delivered”, with UKIs and composefs, from Allison and Timothée at FOSDEM 2025 UKIs and composefs support for Bootable Containers, from Timothée at Devconf.cz 2025 UKI, composefs and remote attestation for Bootable Containers, from Pragyan, Vitaly and Timothée at ASG 2025 composefs backend documentation in bootc Thanks to all the contributors that made this possible, notably (but non exhaustively) from the following projects: bootc & bcvk, composefs & composefs-rs, chunkah, podman & buildah and systemd.

- Announcing Sigrún (Run a command) (2026/04/27 12:15)
  Some time ago I used a feature in KDE called “Run a command” when an event triggered. It triggered for me when a calendar event fired and used Piper TTS to read the event to me out loud. A small popup and a pling don’t work for me. I tried to get the feature back into KDE, but since the merge request isn’t going anywhere and people don’t give details how to implement it correctly I wrote Sigrun now. It is named after a Norse Valkyrie and is short for Signal Run. It is a systemd service running as a user and listening on DBus signals. Once it finds a configured one, it runs its command. The desktop doesn’t matter. Here is the rule that reads my calendar reminders aloud via kde-tts.py: [[rule]] name = "calendar-tts" [rule.event] type = "notification" [rule.filter] app_name = "kalendarac" summary = "Meeting.*" [rule.filter.hints] "x-kde-eventId" = "reminder" [rule.action] command = "/usr/local/bin/kde-tts.py" args = ["-t", "{summary}", "-d", "{body}"] crates.io/crates/sigrun codeberg.org/cryptomilk/sigrun
- Kdenlive 26.04.0 released (2026/04/27 11:30)
  The Kdenlive team is happy to announce the first major release of 2026. This cycle focuses on stability, interface polish and usability improvements. For the first time in Kdenlive's history, this version includes features implemented by so many different contributors. Our developer community is growing, don't hesitate to join us building a free and open source video editing program that respects the users privacy and provides a tool to democratize communication. In case you missed it, check out the State of Kdenlive to learn more about the project's health and the nifty features coming soon. Monitor Mirroring This feature allows you to mirror any monitor while working in fullscreen mode. It’s

especially useful when working with multiple displays or collaborating with others in the editing room, making it easier to share what you're doing without disrupting your workspace. Effects and Transitions This release improves effect and transition workflow. We improved the logic for luma files to adapt to different project profiles and automatically reload previews of downloaded lumas. We added a dedicated tab in the transitions list to browse luma files and added the ability to drag-and-drop transitions directly onto timeline clips. This release also comes with a new Euclid Eraser transition and a Heatmap effect. We also fixed issues with audio TAP effects and improved internationalization support, fixed a bug in the Video Noise Generator effect and scaling issues in the Transform effect. Animated Previews Transitions now include animated previews, making it much easier to visualize how they will look before applying them. Additionally, dropping a transition onto the timeline can now automatically adjust its duration to match the clips above and below, saving time and reducing manual tweaking. The first version of this feature was originally written by Swastik Patel, during KDE's SoK 2025. Automatic Adjustment Transitions automatically adjust to the length of the clip they are being applied to. Math Expressions Added basic math expression support in effect spinboxes. New Heatmap Effect Added a new frei0r heatmap0r effect. Interface and Usability Improvements Timeline This release comes with many usability and workflow improvements to the timeline, such as a Disable Timeline Effects function to timeline hamburger menu, the ability to import and add clips directly from the timeline context menu with a smart length detection function, and sequences now have audio thumbnails. Other highlights include: Continuous Panning Hold the middle mouse button and drag to continuously pan the timeline even when going outside of the screen edges. Implemented by Abdias J Moya Perez. Fixed Playhead Added the option to lock the playhead at the center of the timeline during playback, scrubbing or seeking. Implemented by Abdias J Moya Perez. Playhead/Mouse Zoom Added a button in the Status Bar to toggle between zooming to the mouse cursor position or the playhead position. Implemented by the mrfantastic. Multi-Clip Speed Changes Added the ability to change the playback speed of multiple clips at once either by directly ctrl + dragging in the timeline or by using the Clip Speed tool. Implemented by Vineet Tiwari. Audio Capture Improved support for external recording devices, now the channel count and sample rate combo boxes only display values supported by the selected hardware. Also added the option to choose the sample format (8bit, 16bit, 32bit, and float) as well as a button to use the hardware's default settings. Subtitles and Speech Tools This release fixes issues with cutting, moving, and saving subtitles, and solves a crash cutting subtitle clips. It also fixes a problem where searching for multiple words in the Speech Editor did not work correctly. We also improved the installation process for the Whisper and SeamlessM4T plugins and updated their requirements. Other Noteworthy Features Added Clear Undo History option in the Edit menu Added HD-ready (1366×768) resolution to project profiles Added Add to Project Bin option to Render Widget to directly add rendered file to the Project Bin Hide mouse cursor when placed over monitor in fullscreen and not moving for 2 seconds Added option to edit a video clip with external program (useful for programs like Gyroflow) Rearranged Marker menu items into groups and added a Delete All Timeline Markers action Ability to directly add clips to folders from the context menu in the Project Bin Added AMF encoding profile for Windows Give back to Kdenlive Releases are possible thanks to donations by the community. Donate now! Need help ? As usual, you will find very useful tips in our documentation website. You can also get help and exchange your ideas in our Kdenlive users Matrix chat room. Get involved Kdenlive relies on its community, your help is always welcome. You can contribute by : Helping to identify and triage bugs Contribute to translating Kdenlive in your language Promote Kdenlive in your local community For the full changelog continue reading on kdenlive.org.

- This Week in Plasma: fanciness in Discover and more power efficiency (2026/04/25 00:00)
  Welcome to a new issue of This Week in Plasma! This week includes an interesting blend of improvements. Lots of visual stuff, so get ready for a ton of screenshots and screen recordings! Notable new features Plasma 6.7 The Kicker Application Menu widget can now be configured to show a

“Recent Locations” item. (Christoph Wolk, KDE Bugzilla #420951) Network connections can now be duplicated. (Kartikeya Tyagi, KDE Bugzilla #499188) There’s now a new “has parent window” window rule you can use to target child dialog windows specifically. (Kai Uwe Broulik, kwin MR #8969) Notable UI improvements Plasma 6.6.5 Dragging a search result for a System Settings page to the desktop now creates a launcher to that page as you would expect. This completes a mini-project to improve dragging-and-dropping things to the desktop that we started a while back. (Antti Savolainen, KDE Bugzilla #500259) Plasma 6.7 Discover now has fancier application page headers with more obvious install buttons! (Oliver Beard, discover MR #1297) Discover no longer disables the “More Information” button on list items for in-progress updates. (Tobias Fella, KDE Bugzilla #431719) Discover now does a better job of handling the rare case where an automatic update to a Flatpak app introduces a compatibility issue you can’t easily recover from. Now it will warn you about this once instead of continuously. (Tobias Fella, KDE Bugzilla #509760) You can now drag favorite apps out of their areas in Kicker and Dashboard to un-favorite them. Kickoff is coming soon, too! (Christoph Wolk, plasma-desktop MR #3665 and KDE Bugzilla #518749) When your laptop is plugged in at maximum charge, doing something that changes the power profile to a non-default one now shows only the power profile icon in the System Tray, and omits the fully-charged battery icon because that part is obvious. (Nate Graham, KDE Bugzilla #518802) Improved some awkwardly-worded labels in System Settings and Plasma. (Philipp Kiemle, plasma-desktop MR #3674 and plasma-workspace MR #6522) Frameworks 6.26 Reduced the amount of blurriness seen in icons throughout QtQuick-based apps using the Kirigami.Icon component when using a low fractional scale factor like 150% or less. (Volodymyr Zolotopupov, kirigami MR #2070) Before: After: Added a search provider for startpage.com, so you can search there from KRunner. (Antti Savolainen, KDE Bugzilla #503976) Notable bug fixes Plasma 6.6.5 Fixed a case where the Plasma Login Manager could crash when connecting and disconnecting multiple monitors while the login screen is visible. (David Edmundson, KDE Bugzilla #519302) Fixed some cases where Plasma could crash at login. (David Edmundson, plasma-workspace MR #6520) Fixed multiple accessibility issues: key repeat not working in the Orca screen reader, and various UI elements not being read properly. (Nicolas Fella, KDE Bugzilla #519143, KDE Bugzilla #519333, and KDE Bugzilla #519217) Fixed a tricky issue in Spectacle that could make large images fail to automatically copy to the clipboard right after the app exits. (David Edmundson, KDE Bugzilla #509065) Fixed another cause of the issue whereby de-focused full-screen windows could sometimes inappropriately appear at the top of the window stack. (Xaver Hugl, KDE Bugzilla #484155) Fixed a layout glitch on System Settings’ Colors page that could make UI elements in the color previews overflow when using some non-default fonts and font sizes. (Akseli Lahtinen, KDE Bugzilla #516413) Changing the brightness or any screen settings no longer terminates Spectacle’s sectangular region recordings. (Xaver Hugl, kwin MR #9127) Frameworks 6.26 Fixed some visual glitches around radio buttons in the Audio Volume widget. (David Edmundson, ksvg MR #103) Notable in performance & technical Plasma 6.6.5 Fixed an issue that made System Settings’ Touchscreen page appear while the “highlight changed settings” feature is enabled even if you don’t have a touchscreen. (Jin Liu, KDE Bugzilla #518868) Plasma 6.7 Turned on the “overlay planes” feature for Intel GPUs, which should improve performance and save some energy when using cooperative games and apps. (Xaver Hugl, kwin MR #8699) Improved power efficiency for full-screen windows and effects that don’t gain any benefit from using the “direct scan-out” feature; now they’ll only use it if it will save power. (Xaver Hugl, kwin MR #9120) How you can help KDE has become important in the world, and your time and contributions have helped us get there. As we grow, we need your support to keep KDE sustainable. Would you like to help put together this weekly report? Introduce yourself in the Matrix room and join the team! Beyond that, you can help KDE by directly getting involved in any other projects. Donating time is actually more impactful than donating money. Each contributor makes a huge difference in KDE — you are not a number or a cog in a machine! You don’t have to be a programmer, either; many other opportunities exist. You can also help out by making

a donation! This helps cover operational costs, salaries, travel expenses for contributors, and in general just keeps KDE bringing Free Software to the world. To get a new Plasma feature or a bug fix mentioned here Push a commit to the relevant merge request on invent.kde.org.