Guix Foundation 2026

Fundraising & Future

Guix Foundation Intro

- The GNU Guix project is self-funded - 95% of contributors are volunteers
- Guix has two fiscal Sponsor Organisations:
  - FSF
  - Guix Foundation
- You can Donate to Guix through either

A 'fiscal sponsor' collects donations, and holds assets on behalf of the project.

- Previously there was Guix Europe Foundation
- This year we became **Guix Foundation**
  - Reflects that we want to support GNU Guix globally
  - Our priorities and focus are driven by our Members
  - We changed banks and added platforms for donations
- Thanks to **Simon Tournier** and **Tanguy Le Carrour** for the transition to Guix Foundation - they did a lot of work!

Sustain & Strengthen Campaign

- After four months of fundraising we've raised **€11378**
  - This puts us at 75% of our €15000 target
- 136 people have stepped forward to become **recurring donators**
  - We estimate **€1650** of donations per month (annualised **€19800)**
  - Nov-January 17 new people started, 8 stopped
- Donations through the FSF **additional** to these figures (I don't have visibility)
- There were some big donations from people - amazing!
- **Thanks to everyone who took part - really fantastic!**
- SUSE Cares are donating $500 (~ **€400)** our first donation from an organisation/institution - thanks SUSE

We estimate that we'll achieve €33,900 of funding

Sustain & Strengthen Campaign

- Recurring donations are **critical** for our sustainability
  - Most of our expenses are are regular - hosting and build-farms
  - A regular stream of donations means we can align those expenses with our donations
- **Guix Days** registrations are **€3830**
  - Multiple levels that enable choice seems to have gone well
  - A number of people supported at the higher-level - thank-you!
  - It's really helped Guix Days to be sustainable
- There are some lessons and a fair bit of complexity to manage for Tanguy/I
- Will end the campaign in the next few days with a blog post

Donations	October	November	December	January	Total
Donations	€3574	€2801	€2468	€2535	**€11378**
Guix Days	€0	€1845	€1720	€265	€3830
Total	**€3574**	**€4646**	**€4188**	**€2800**	**€15208**
Fees	(€204)	(€282)	(€243)	(€226)	
Net	€3370	€4364	€3945	€2574	**€14253**

Guix Foundation Budget

- How much should we spend?
- How should we spend it?

Budget Overview (revenue)

- Draft budget and forecast for 2026
  - **May change - depending on review from Tanguy and SAC**
  - The SAC is a small committee of members that makes all decisions
- For these slides it's split into revenue and expenses

- Best to be reasonably conservative about revenue
  - For example we built in some buffer in case donations slow down

Revenue	2026	Notes
Guix Foundation memberships	€800	Last year was over 800
Fundraising campaign	€11000	'Cash' we've received
Recurring donations (forecast)	€13200	Estimate done on 9 months
Guix Days registrations	€3500	
Misc	€400	This is the SUSE Cares donation
Transfer from FSF Fund	€5000	Aiming for more than this
SUBTOTAL	**€33900**	

Budget Overview (expenses)

- Expenses are forecast somewhat generously
  - Better to forecast too much than too little
- We keep a contingency reserve for critical expenses - we will try and build this further using existing funds and FSF transfers
- Forecast (not shown) to leave the year with at least €13000 of cash

Expenses	2026	Notes
Hosting and infrastructure	€5812	
Conferences	€5250	
Community & Promotion	€500	
Development	€15000	
Operations / G&A	€1350	
SUBTOTAL	**€27912**	
Contingency reserve	€4250	A reserve for critical expenses (i.e hosting)
Total budgeted expense	€32162	
Retained (vs revenue)	€1738	Small additional retained

Hosting and infrastructure

This category covers both project infrastructure and developer experience. A great development experience is important for Guix's sustainability so that happy contributors continue. Over 2022-24 there were 454 contributors, with the majority (60%) being 'casual' (less than 20 patches).

"We really need to improve the CICD situation. I see we have so many system tests that catch issues. Let's make sure each patch has run at least a couple of those system tests before it is being merged, or even before a reviewer has even looked at [it] ..."

"Minimizing the required work needed to keep packages up to date ..."

2026 Budget funds:

- Current build-farm hosting and hardware
- **Upgrades to storage** (agreed) which should improve builds
- Hosting Web and other supporting services (Hertzner)
- **Supporting Codeberg (proposal)**
  - Critical for our Git repo - developers and users

This is our critical category, we retain a contingency budget for an emergency.

Community and Collaboration

This category covers ways in which our existing users and developers can come together to communicate and collaborate. As a highly distributed community we're dependent on both technical and social relationships.

In the past this was for Guix Days, but ideally we would do more than that. It would be great to have other developer meetups whether large or small.

2026 Budget:

- Guix Days
- Potential budget if people want to organise some small, local dev meetups
- **(proposal) to create a GuixConf**
  - Online user / developer conference
  - Goal to both be for people 'using' Guix and developing Guix
  - Fabio and Steve - looking for other people to help - you?
  - An event on a weekend - presentations and discussions
  - Calendared for June

Advocacy and Education

To have a healthy, sustainable and active project we want to attract new users to Guix. This means educating, advocating and promoting Guix in other communities.

Many projects do this by members promoting by attending Events, Conferences and running stands or doing talks. The role Guix Foundation can play is by connecting our members together and supporting these activities.

2026 Budget:

- **Opportunity for members to present proposals to ...**
  - Run a stand at an Event
  - Do an installfest
  - Give a talk in a tech community
  - Teach Guix to students

Guix Development

Guix Foundations goal is to "develop software and operating systems" specifically Guix. We do that indirectly with infrastructure, we can also directly **accelerate development**. This would be in the form of grants or similar mechanisms.

We know that other communities such as Codeberg, FreeBSD, and Clojure have accelerated development through small stipends / grants. Some Guix developers have already had grants, such as those through NLNet.

2026 Budget:

- **(proposal) Experiment with a Grant system**
  - Modelled on Clojurists Together
  - Light-weight proposal from people seeking a grant
  - Decisions focus on high-impact areas identified in the community - these come from the User Survey
  - Only 'ask' is that people provide a monthly report that's published on what they've done - high trust and low admin
  - Grants will be small - it's partial support - given our size as a community

Foundation Operations

We've demonstrated that we can raise money for Guix - **thanks to everyone who's helping out!**

Running an Association legally and dealing with the banking system is complicated. **Tanguy is an unsung hero**, spending hours and hours a month working through all the admin and accounting - he deserves our thanks!

The Foundation has seen a lot of growth with new members, we want to continue that and also **get members involved**! It should be a truly members-driven organisation.

2026 Budget:

- **Improve communications**
  - Revise the Web site
  - Add budgets and reports to the site so it's transparent
  - Update members regularly - monthly email
- Add additional funding platforms
  - Adds complexity for Tanguy / I - but reduces friction for donators
  - One example is donations using banking in North America
- Annual General Assembly (online)
- Explore other funding options - sponsorship from organisations / grants (stretch)

Your views?

Guix Foundation relies on individual people stepping forward - we want to make sure that we use the funds they've donated to improve Guix in a way that helps them. Guix Developers are a key constituency, without their hard work there is no Guix for users. And, many of us fall into both camps! It's a virtuous circle if we get it right.

Maybe a discussion for Guix Days?

- What could Guix Foundation do that would help users the most?
  - The **2025 User Survey identified key problems** (keep going in these slides)
  - https://slides.com/futurile/survey-says-5-guix-dev-priorities
    - 1. Improve speed of review
    - 2. Package freshness
    - 3. Firmware
    - 4. Runtime performance
    - 5. Errors and debugging

- What could Guix Foundation support that would help Guix devs the most?

- Are the categories and ideas presented the right ideas, what else?

Thank You!

5 Guix Dev Priorities

What the survey tells us the priorities are for adopters, users and contributors

(ideas for discussion!)

#1 Improve review speed

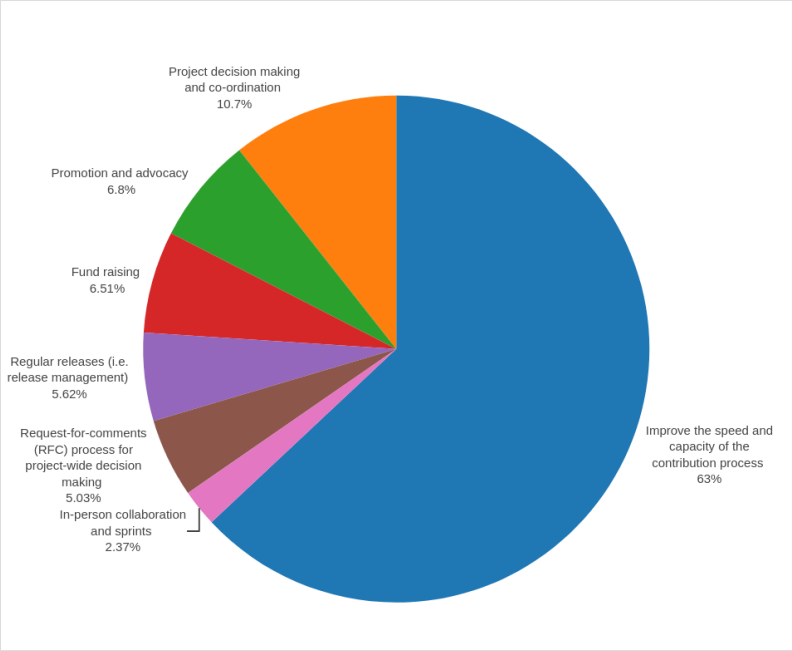
#2 Package freshness

#3 Firmware experience⚠️

#4 Runtime performance

#5 Errors and Debugging

#1 Improve review speed



- Timely reviews with actions taken on contributions is **overwhelmingly** the biggest request from contributors
- Over 60% of developers selected it as their highest organisational request. It was top-3 in every question!
- Directly damaging morale of contributors

"I really dislike that 70% of my patches don't get reviewed at all, how simple or trivial they may do. I do really test them and dogfood so contributing seems like a waste of time as someone without commit access"

"already checked "timely reviews/actions", but I want to emphasize how demoralizing my contribution experience was. I was excited about how easy it was to make, test, and submit a patch; I would love to help improve the packaging situation for things that I use. But it's been about a year now since I submitted some patches and have received exactly 0 communication regarding what I submitted. No reviews, no comments, no merge, nothing. Really took the wind out of my sails"

"Slow, or sometimes inexistent, feedback for submitted patches and issues"

"I still use Guix, but am a previous contributor. Important patches (for me) which I submitted were/are ignored, so I've stopped contributing

#1 Improve review speed

Social and organisational focused comments noted that the project has more work than the current committers can handle. A consistent suggestion in the survey is to **add additional committers**.

"Keep manual up to date, I think we need more committers doing reviews and giving more love to darker corners"

All the best. The project might need more hands to review incoming patches"

If you ask 900 people their opinion you get passionate views. We should be aware there were some negative comments about the project being elitist and inward looking:

" I was passed over for commit access (even though I surpassed the 50 commit requirement) because I could only find 2 people to vouch for me, not 3. Then my patches stopped being merged, and some 2-year-pending patches I sent were closed without good reason. With the way Guix is run and how they treat contributors, it is an insulting/degrading process that I am no longer willing to put myself through."

Q: How can more committers be added? What prevents this?

#2 Package freshness

The survey showed that users and contributors want the latest packages (package freshness). Adopters identified missing packages as being a key reason why they couldn't use Guix.

"Much of the software I needed wasn't packaged, and it eventually became frustrating. I tried to package what I could, but some things felt extremely difficult, E.g., `jujutsu` `ghc`. [continues ...]

"To many packages updates were lagging behind, this was raising concerns for me from a security point of view"

Ideas for discussion:

1. Do the things in #1 that speed up reviews to increase capacity.
2. Add a community repository**:** in a similar way to the Arch User Repository (AUR) which is more open access. New developers can build up experience in this environment. Users choose to enable this if they wish. Core committers can promote from here if they wish.
3. Add a pull request workflow. This would increase the velocity and ease the path for new contributors.

Q: If we can't increase the number of committers, can we change the framework by adding a community repository?

#3 Firmware experience⚠️

Firmware not being part of the distribution is a big problem for both new adopters and long-term users: 33% of users said it limited their satisfaction.

"*Exclusion of all references to non-free software (and no suggested step-by-step easy setup) made a full-featured initial installation untenable.*"

Doing something in this area will be difficult due to the GNU FSDG. The survey also showed strong opinions on all aspects - our community is a wide tent.

Ideas for discussion:

1. Improve documentation so users are at least informed about Nonguix.
2. Provide Nonguix as an option in the installer and let them choose. 60% already do!
3. Create our own equivalent of Debians rules (DFSG) and allow firmware as they did. In Debian's case they created a separate archive and include it in their installer.

Q: Given the constraints what can be done to help users? What is the best for users and Free Software?

#4 Runtime performance

"Foremost faster guix evals. I forget what I was doing while it runs"

"Building guix takes too long time for packagers. It is not clear why everything needs to be compiled when only contributing a package. Why does the documentation need to be build when adding a package?"

Improving Guix's runtime performance for contributors centres on how long it takes to build and test packages.

"The core tooling was far too slow (e.g. pulling updates, etc.); Nix is slow, but nowhere near as slow as Guix (was back then, but I'm not aware of the kind of order of magnitude improvements that would have been required). Core functionality was not reliable enough for a server operating system (shepherd, logging, system rollback). [continues]"

"Guix pull is too slow. The guix ci servers are inaccessible from my location, requiring a proxy."

Improving Guix's runtime performance for users is wrapped up in their perception that "guix pull" takes too long. There are also mentions of how much resource Guix uses on small systems. It impacts about **50% of users**, so it's a big issue. Is this problem tractable?

#5 Errors and Debugging

Contributors **highest priority** technical improvement request is debugging and error reporting. About 20% of contributors asked for this, and it's ranked in the top 3 overall.

"I just want to reiterate that the debugging process can be painful sometimes. Sometimes guile gives error messages that can be bewildering.

"guix is very-very slow and consumes too much memory and CPU for what it's doing. also error messages are the worst i've seen in my 10 years of programming"

Generally for contributors that are new to Lisp and/or new to Emacs the tools are clearly a big issue. Debugging issues with tooling is also problematic since Guix is a completely different approach. Together these create a high barrier.

There weren't any particular suggestions just lots of "this is hard!". In the widest sense there were requests for more 'how-to contribute' guides.

Similar to Runtime Performance, is this a problem that can usefully be focused on?

How do adopters struggle?

1. Shortage of example configuration: templates and code to use
2. Lack of practical guides, how-tos and videos: not an issue with the manual
3. Lack of integrated drivers: not knowing about nonguix or unable to configure
4. Missing packages or out of date packages
5. Unable to run compiled binaries: binaries compiled on a "standard" Linux
6. Issues with encryption
7. Runtime performance and efficiency: guix pull too slow and using too much resource
8. Complexity of learning curve and maintenance: too much overall effort required
9. Nix is more practical, featureful or popular: e.g. missing 'flakes' features.
10. Language ecosystem issues (Docker, Go, Rust, PHP etc)

Q: There's inherent complexity in Guix as it represents a different approach with substantial benefits. Even for expert Linux users it's a difficult learning curve. Additionally, there's accidental complexity in the implementation. **How could the tooling, docs and support systems help users adopt?**

Guix Users

- How does usage develop?
- What do users love?
- What do they struggle with?

Guix System Usage

- Over 70% of users use Guix as a graphical desktop **directly on hardware**
- Server usage jumps from 5% of adoption to around a third
- Guix home is used by Guix System users, but not by hosted users (17%)
- A lot of users are packaging their own software - comments compare vs Nix flakes

Usage type	Use %	Stop %
Guix package	64	17
Guix home	48	9
Guix shell	36	10
Package own software	40	9
Deploy (deploy/pack)	19	8

Deployment type	Percentage
Graphical desktop on hardware	73%
Server on hardware	24%
Server in a VM	18%

Q: What are the implications for packaging priorities, testing and docs?

Q: How can Guix make it easier to on-board into some of the features? (e.g. Guix home) Build on Guix shell? Why are the deploy features lagging when server usage is high?

Hosted Guix Usage

- Possible difficulties in the user experience when hosted as the stop % are much higher (e.g. nearly 20% stop using Guix Home.)
- A lot of comments about difficulties using (graphical) Guix packages when hosted

Usage type	Use %	Stop %
Guix package	50	26
Guix home	17	11
Guix shell	41	18
Package own software	28	9
Deploy (deploy/pack)	13	7

Q: What does 26% "stop" for guix package mean, using Guix home?

Potential for expanding the 'package my software' use-case?

"*Guix home breaking Fedora. Troubles with binary applications due to the non-fsh nature*"

"*Problems using it on a foreign distro. Guix Home particularly assume that you are using guix system. I had to tweak the .profile a lot to get it working.*"

I found setting the numerous variables and comments recommending I do so contradictory and so confusing"

"Some DEs don't integrate as well as they do on other distros."

Satisfaction & Limiters

- Most users are satisfied and ❤️ Guix
- Survey asked which areas limit users satisfaction

70% satisfied

User satisfaction limiter	Percent
Guix runtime performance (e.g. guix pull)	48%
Missing or incomplete services	40%
Error messages and debugging	39%
Shortage of informal guides & examples	39%
Hardware drivers not included	33%

"*examples, a show of how a task is done in Debian or OpenSUSE and contrast it with how the task is done is guix would be helpful*\"

"*guix is very-very slow and consumes too much memory and CPU for what it's doing. also error messages are the worst i've seen in my 10 years of programming*"

"*coming from Nix: smaller, less up-to-date-package set, substantially fewer home services*"

"*Guix is unusable without nonguix / proprietary drivers*"

What should Guix improve?

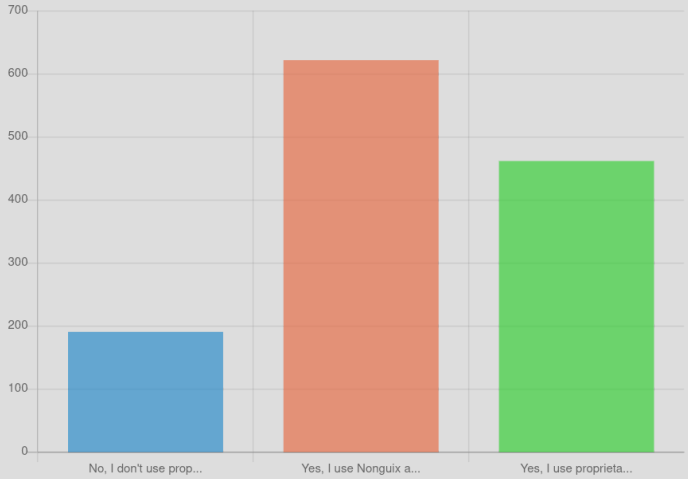
Rank 1	Priority
Package freshness (latest vers)	1
Performance - faster guix pull	2
Make Guix easier to learn	3
Package reliability	4
Hardware drivers	5
More packages	6

All Ranks	Priority
Performance - faster guix pull	1
Make Guix easier to learn	2
Package freshness (latest vers)	3
More packages	4
Package reliability	5
Hardware drivers	6

- Performance and resource usage is consistently in the top 3, with many comments.
- Package freshness and more packages are consistent as something users want, and the lack of required packages prevents them from using Guix more. Users and contributors consistently rejected a smaller distribution with more focused packages.
- Package reliability: many comments are about configuring and using packages - a missing service - or not working (graphical apps) in a hosted environment
- Docs means how-tos, example code that can be cut-n-paste and videos

Hardware drivers

- Most users use Nonguix and drivers when using Linux



66% use Nonguix

"As a long time Arch user I found it difficult to configure Guix for daily use. I need proprietary video drivers (and possibly other bits to get everything working?) and I don't remember if I ever got those up and running."

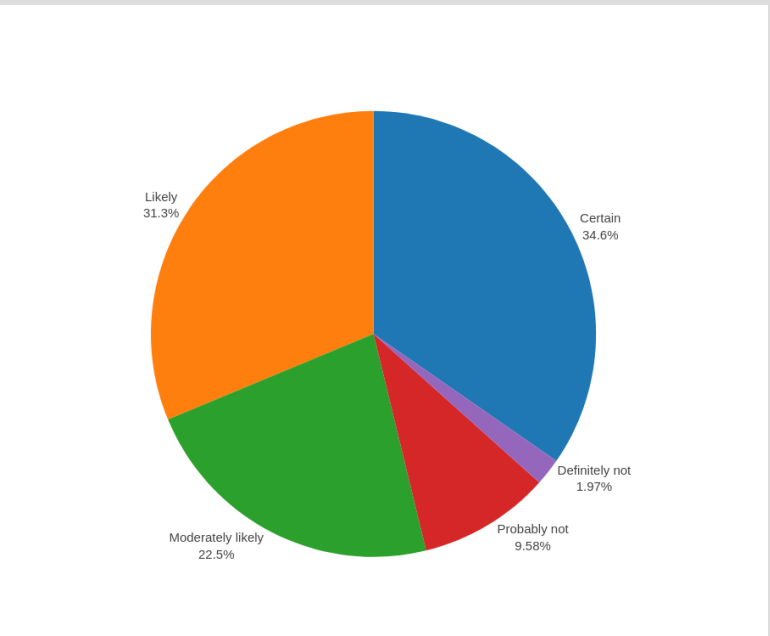
"Exclusion of all references to non-free software (and no suggested step-by-step easy setup) made a full-featured initial installation untenable."

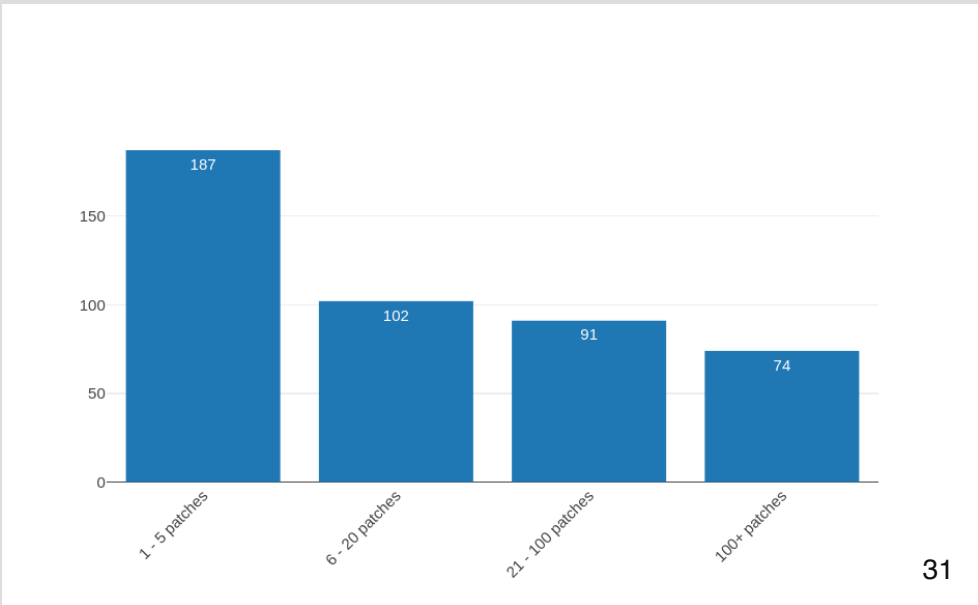
"Guix is unusable without nonguix / proprietary drivers"

Guix Contributors

How do people contribute?

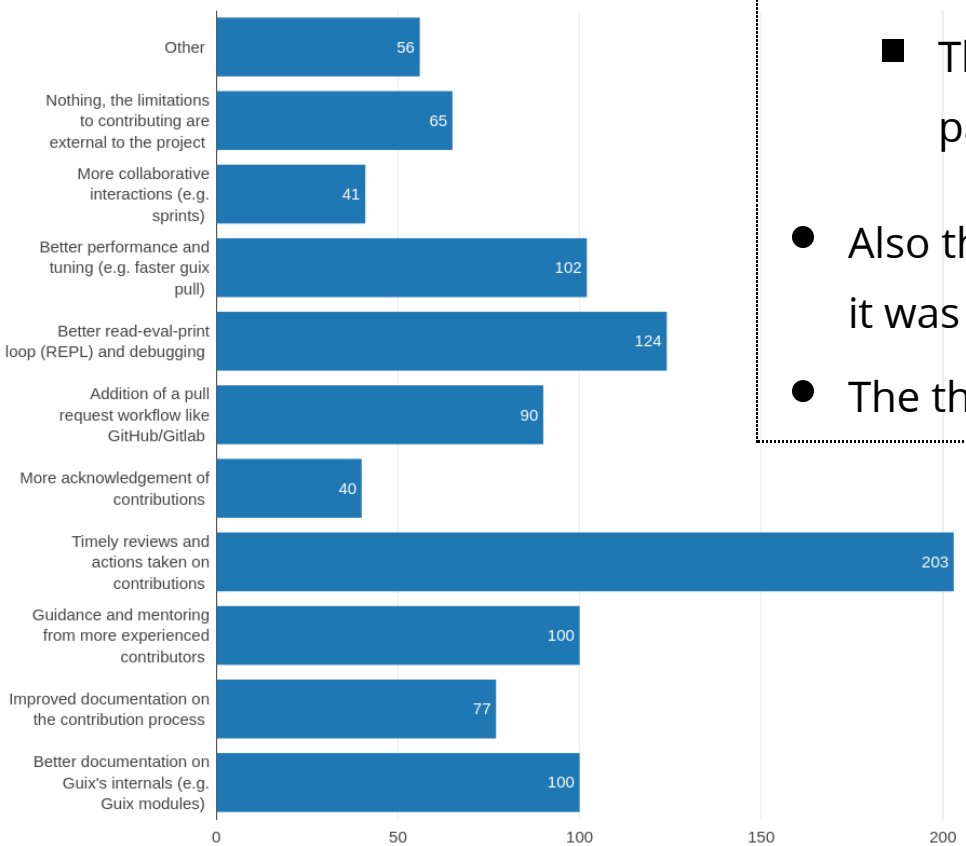
- Around 454 people sent patches in the last two years ('active contributors')
- Many sent just 1, or a few patches. The top-10 contributors have all contributed over 700 patches each
- Most contributions are code (60%), but there's also plenty of other things happening!
- Project is 94% volunteer driven
- Contributors are reasonably satisfied - about 35% are Certain to contribute again





Contribution friction

- Previous contributors (59 participants) comments are very interesting. The two things that put them off are:
  - Response to contributions was too slow
  - The contribution process (e.g. email and patch flow)
- Also the number #1 issue for active contributors it was P1 on every question
- The theme is lack of response and a lot of friction



16. Check list for contribution

https://guix.gnu.org/manual/en/html_node/Submitting-Patches.html

1. Each patch contains one discrete change only: changes to one package
2. Package source has no bundled libraries (e.g. vendored libraries)
3. Verify the source and ensure the Package hash is correct (Step 8)
4. Package builds correctly (Step 10 & 12): check the source is downloading and that it builds deterministically
5. Package works (Step 11): test it in a guix shell
6. Package builds on other architectures: Guix's CI does this
7. Patch commit message meets the standard (Step 13): look at previous commit messages
8. Package only references other packages it needs (Step 14): also guix size
9. Package is styled correctly (Step 15): guix style
10. Package Synopsis and Description are adequate: some maintainers want factual text (a bit subjective!). Guix lint checks things like spacing and no use of definitive article
11. Package passes lint (Step 15): guix lint
12. Patch goes to the topic branch of a team if possible: e.g. rust-team or python-team
13. Packages for the master branch don't involve substantial rebuilds: new branch potentially

Technical improvements?

Rank 1	Priority
Debugging and error reporting	1
Package freshness (latest versions)	2
Automate patch test and acceptance	3
Runtime performance (speed / memory use)	4
Package reliability (installs and works)	5
Contribution workflow (e.g. Pull Requests)	6

All Ranks	Priority
Automate patch testing and acceptance	1
Runtime performance	2
Debugging and error reporting	3
Project infrastructure (e.g. CI)	4
Contribution workflow (e.g. Pull Requests)	5
Package freshness (latest versions)	6

- Contributors #1 request is to improve the speed and capacity of patch reviews
- Package freshness and more packages are consistent as something users want, and the lack of required packages prevents them from using Guix more. Users and contributors consistently rejected a smaller distribution with more focused packages.
- Package reliability: many comments are about configuring and using packages - a missing service - or not working (graphical apps) in a hosted environment
- Docs means how-tos, example code that can be cut-n-paste and videos