Why Lix?

(page under construction. editor's note: parts of https://pad.lix.systems/lix-manifesto (PRIVATE) are ported, parts need review before posting here)

(editor's note (ii): this page wants to be a contributor facing page, as opposed to the website page that maybe will have more general info?)

We should introduce ourselves! We are the Lix team, and we are working on a fork of CppNix focused on stability and user experience over features.

Core team members

- puck (@puckipedia), she/her
- hexchen (@hexchen), she/her
  hexchen is working primarily on mantaining and extending the Lix project infrastructure.
- Qyriad (@Qyriad)
  Build system experts who delve way, way too deep into tooling
- eldritch horrors (FIXME(horrors): github if desired?), they/them
- wiggles (@9999years), she/her
- Irenes (@IreneKnapp), they/them
- jade (@lf-), they/them
  jade is working on packaging, testing, infrastructure, tooling, review, stability, and a large amount of the writing in Lix. They are currently studying Computer Engineering at UBC in Canada.
- raito (@RaitoBezarius). he/they
  Raito is working on nixpkgs packaging, infrastructure, review in Lix.
  They are a Tvix developer focusing on the store and the evaluator.
- Kate Temkin (@ktemkin)
  A performance art piece written live by a collective of hardware hackers & low-level engineers. Kate works on Lix as part of a commitment to helping you do cool things, and is seriously considering rewriting every bit of documentation ever to cross paths with Nix.
- Lunaphied (@lunaphied), she/her (singular), they/them (plural)
  Lunaphied spend a disproportionate amount of their time considering how to get FPGAs as far from Earth as possible. When they’re not working on Space Stuff, they consider doing the same for Nix regressions.

FAQ

What is Lix anyway?

Lix is a fork of CppNix 2.18, focused on stability and the user experience of both users and contributors. We want to create a safe platform to move Nix technology forward, as a piece of critical infrastructure.

To this end, we have instituted a freeze on the core, where we apply high standards to changes to the core of the system and pursue testing and stability as our first priority on the core. Our long term vision is to shrink and decouple parts of the core, and move features like Flakes to the periphery of the system.

To achieve our goals in user experience, we are allowing significantly more contributions, still with tests, to the user facing surface of the system where there are fewer stability guarantees, and explicitly define what is expected to be stable and what can change.

Part of our work on the interface of Lix is in Qyriad's project Xil, which is an experiment in an alternate CLI for Nix implementations, which will potentially slowly merge with the Lix CLI.

Technical differences from CppNix

- Lix is built with Meson, so language servers will just work on it
- Lix does not include lazy trees, and does not intend to use the upstream implementation of lazy trees; something like lazy trees is planned (FIXME: publish the planning document for that).
- Lix does not use libgit2 and does not intend to use it
- Lix is entirely self-hosted in terms of infrastructure and uses Gerrit/Forgejo instead of GitHub
- `nix repl` can `:doc` library functions
- `nix repl` can accept overlays as config files; see `repl-overlays` release note in the sources
- Performance improvements (8-20% faster than 2.18)

Views on flakes

The Lix project acknowledges that flakes are the way that the majority of people use Nix today, and does not intend to remove support for them. However, as part of our overall focus on stability and dependability, some features of Flakes will be changed to be stricter.

Flakes are not the only way to write Nix language code in Lix, and we intend to provide a good experience to flakes users, while also improving the experience for those not using flakes, by evolving a compatible but more flexible flake-like abstraction in the periphery of the Lix system.

Why is Lix different from tvix?

tvix is a Nix implementation from the ground up in Rust, aiming to be compatible with CppNix, by building a system from the ground up. It is developed by some of the same people. tvix also aims to improve the stability of Nix technology, but with the approach of starting from the beginning.

Lix is intended to evolve CppNix into a stable foundation for future evolution, without breaking clients along the way. Its goals are to aggressively pursue technical debt and remove the skeletons from the closet, while remaining deliberate about behavioural changes through testing. Lix will contain Rust components in the near future.

The two projects have similar goals but different approaches, and there will likely be cross-pollination between them; though cross-pollination of code is difficult due to licensing.

Revision #9
Created 25 March 2024 21:30:55 by jade
Updated 5 May 2024 23:49:31 by Qyriad