Freezes and recommended contributions

Suggested contributions

Consider taking an issue marked E-help wanted: assign it to yourself and have a go. Feel free to ask for help in the development channel. The Lix team wants these issues fixed, but they are not high on our agenda to fix ourselves.

When in doubt, please ask the Lix team before beginning work, to make sure it is in line with our current priorities.

I don't wanna write C++. How can I help?

This is totally reasonable, C++ is not super fun, but this isn't to say there isn't help if you do choose to. Nonetheless, there are still many issues on our tracker that do not require writing any C++ that would help out immensely.

Here's some possible places to look:

- docs issues on lix-project/lix. These generally involve writing documentation prose.
- testing issues on lix-project/lix. A large number of these are just writing shell scripts, pytest scripts (`tests/functional2` in lix), or fixing NixOS tests
- external project issues. This is a veritable grab bag of issues in external projects that we would love to have fixed. There's some in Gerrit (Java), some in Forgejo (Go), some in meson (python), some in nixpkgs. It's unlikely these will get done without some help so we really appreciate help!
- website issues our homepage still need some love in various areas. It uses Hugo as a static site generator, so contributing requires little more than some basic HTML and CSS skills.

New builtins

One thing that is a little bit tricky and can get contributions canned late in the process is new builtins. We should write up a more full document on this, but the gist is that the API of any builtins needs to stand the test of time as they have pretty strong compatibility and usefulness expectations.

As such, it would be greatly appreciated if work on builtins starts with an issue on lix-project/lix tagged RFD discussing the use case for the builtin, then, once there is rough agreement on the use case, write a simple document giving examples and design of the proposed API. Then, we can gather feedback before too much time is put into implementations that might not see the light of day.

Freezes

We expect to have `main` always be in a state to run on machines you care about, unconditionally. Nightly builds should not be a problem to run in production in any freeze state.

For the time being, Flakes are frozen in their current feature set and will only receive maintenance bug fixes.

Revision #13
Created 2024-03-25 10:35:50 UTC by jade
Updated 2026-02-01 13:14:00 UTC by piegames