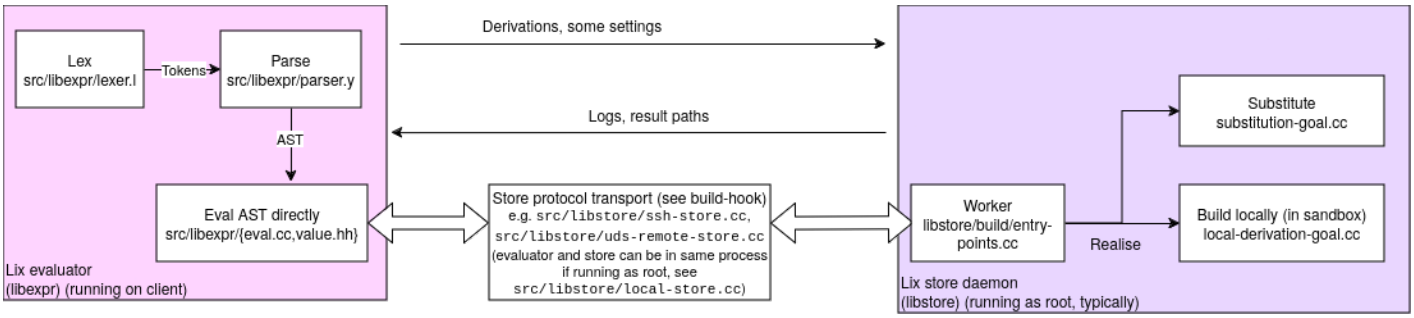
Codebase overview

The Lix system is constituted of two broad parts, the evaluator and the store daemon. The two pieces may run on the same machine or on different machines.

For example, in a remote build setup like https://hydra.nixos.org, one node is running several evaluators in parallel, and builds are dispatched to several builder nodes.

(fyi to anyone editing this, double click the image in the preview to edit it)



Evaluator

The evaluator is an AST tree-walking evaluator with lazy semantics.

Notable files:

- libexpr/value.hh, which defines the interface for evaluated values' interactions.
- libexpr/eval.cc, where most of the evaluator is.
- libexpr/nixexpr.cc, where the most of the `nix::Expr` class hierarchy is implemented, which are the AST types for the evaluator.
- libexpr/primops.cc, defining builtins.
- libexpr/parser.y, the (current) yacc generated parser.
- libexpr/lexer.l, the bison-generated lexer.

Known design flaws

- GC issues (FIXME add details)
  - General tendencies to leak memory. Hydra restarts the evaluator every so often if it runs out of memory.
- AST based evaluator design limits perf
- Stack tracing has issues that make the traces confusing (FIXME add details)
- Funniness with attr ordering and equality that nixpkgs depends on, which is fragile

    - Currently no real tools to diagnose this and stop nixpkgs from depending on it.

      https://github.com/NixOS/nix/pull/8711 exists but regresses perf a lot and is not mergeable.
- Evaluation-time build dependencies (often called IFD) block the evaluator rather than allowing other evaluation to proceed
    - This has significant downstream effects such as typical derivations building hand-written large pieces rather than generated smaller pieces with IFD, since IFD is bad.
- The eval cache both has false hits and false misses, and needs redesign.

Lix team plans

- Rewrite parser (done, being ported for 2.91 by horrors)
- Rewrite evaluator to be amenable to moving to bytecode (horrors) (long term)
- Do something about GC (long term)

Store protocol

The store protocol is a hand rolled binary protocol with monotonically increasing versioning. It runs over a few different transports such as ssh (`src/libstore/ssh-store.cc`) or Unix sockets (`src/libstore/uds-remote-store.cc`).

Known design flaws

- We cannot extend the store protocol (not that it is Good) because of the monotonic version numbers: we must always be stuck at *some* released CppNix version. This significantly moves up the need to replace it.
    - The code is significantly tangled with the current protocol design.

Lix team plans

- Replace protocol with capnproto, transport with websockets?
    - Would likely be in addition to existing protocol; existing protocol likely would be run through a translator.

Store daemon

The store daemon takes derivations (≈ execve args and dependencies) and realises (builds or substitutes) them. It also implements store path garbage collection.

Lix's local store implementation currently uses a SQLite database as the source of truth for mapping derivation outputs to derivations as well as maintaining derivation metadata.

Notable files:

- libstore/build/local-derivation-goal.cc, which implements the local machine's builder including the sandbox
- libstore/build/entry-points.cc, the server side entry points of the store protocol
- libstore/daemon.cc
- libstore/uds-remote-store.cc, the client implementation of Unix socket stores

Known design flaws

- Sandbox is of dubious security especially on Linux (where it is actually expected to be somewhat secure)
  - Overall tangled code around the sandbox, particularly in platform specific parts
- Poor self-awareness: daemon doesn't know what it is building
  - Due to this plus the protocol being frozen, it would be very hard to implement e.g. dropping into a shell on failed builds
- Substitutions are inherently a kind of build so they can't happen out of dependency order or with better parallelism
- SQLite database has a habit of getting corrupted (probably due to Lix-side misuse)

Lix team plans

- Replace sandbox with other software, perhaps bwrap
- Fix daemon self-awareness, add protocol level features to make this better
- Rearchitect substitution to enqueue weakly ordered jobs that happen in parallel and can resume downloads
- Switch to xattrs as the source of truth of store path metadata such that the SQLite DB can be completely rebuilt

---

Revision #6
Created 2024-04-03 11:22:54 UTC by jade
Updated 2024-04-17 13:53:55 UTC by jade