GSOC'24: Nix Internals: Use std::filesystem::path for Path

Siddhant Kumar

March 18, 2024

1 Motivation

Personal motivation

I've been a NixOS user for a long time and I always wanted to learn more about the project and possibly become a contributor. For me, the GSOC program is an excellent way to get mentorship from an experienced programmer and learn about how projects at this scale carry development.

My exposure to C++ has primarily been academic, limited to problem-solving on platforms like Leetcode and Codeforces. Contributing to this idea would not only enhance my understanding of Nix's internals but also deepen my comprehension of how complex projects effectively utilize C++.

Project motivation

The Nix project already uses some C++17 features like `std::optional` and `std::string_view`. Incorporating the use of `std::filesystem` would encourage the use of the standard. It'll also reduce the source code size reducing the maintenance effort.

It will also aid in porting Nix to Windows, since an explicit design goal of std::filesystem is to be portable between Unix and Windows.

- Relevant issues on NixOS/nix:
  - `https://github.com/NixOS/nix/issues/9205`

2 `Path` usage in nix

- `Path`, defined in `libutil/types.hh` is referenced in 382 places[1] in the codebase (including tests). The usage is spread across the codebase with operations like ”joining paths” that are currently Unix only.
- `PathView`, `Paths` and, `PathSet` also defined in the same header file is referenced 56 times mostly in function parameters.

3 Suggested approach

An approach to carry the porting process is described below.

Analyse the usage of `Path` in the codebase

It is crucial to discuss which parts of the code need changing and what can remain unchanged. The codebase utilizes `CanonPath`, which is Nix's representation of a file path that is independent of the filesystem. Code that does not interact with the real filesystem should employ `CanonPath`.

We begin by examining the usage of the `Path` type alias. This can be accomplished by replacing `Path` typedefs with `std::filesystem::path`.

Executing this change will result in numerous compilation errors, which we can utilize to guide further modifications.

Alternatively, we could use editor tooling to locate references and analyze them grouped by libraries (e.g., ‘libutil‘, ‘libexpr‘, and ‘libstore‘).

Port utils declared in `libutil/file-system.hh`

This file (and the corresponding .cc) include a lot of utility functions for filesystem operations that are spread across the codebase. The work will start by updating these utilities to use `std::filesystem` where possible.

Update remaining uses of `Path` in the codebase

Once the utilities are updated, we'll still have plenty of code that refers to the `Path` typedef. However, most of these references involve simple tasks like reading a path, combining two paths, or using utilities declared in `libutil/file-system.hh`, which we've already updated. Any remaining uses of `Path` will be manually reviewed and updated if needed.

4 Timeline

The GSoC 2024 program has some flexibility in the schedule for projects. The length of time allowed to complete a project can range from 10 weeks to 22 weeks for medium and large projects with the standard length of 12 weeks.

This is just a rough estimate which is very likely to change after discussions with the mentor and the coding time may vary depending on the coding complexity.

- **May 1 - May 10** Discussion with the mentor with the goal of finalizing the changes.
- **May 11 - May 20**
  1. Setup a base PR that will be used for writing reports, discussion on goals and merging in smaller TODO PRs.
  2. Learn more about the codebase and C++17
  3. start coding.
- **May 21 - July 8** Coding - port fs functions in ‘libutil‘ to use `std::filesystem::path`
- **July 9 - July 12** Midterm evaluation submission includes ported ‘libutil‘ and relevant tests
- **July 13 - August 19** Coding - port the remaing uses of `Path`, `PathView`, `Paths` and, `PathSet` to `std::filesystem::path` or `CanonPath` where necessary
- **August 20 - August 26**
  1. Get the base PR ready for merge to master, resolve reviews, add missing tests if any.
  2. Final report submission

5 Past experience with nix

As a contributor

I’ve made one PR[2] in nixpkgs, which was simply a version change for an already packaged binary (Heroic Game Launcher). Other than this, I don’t have any past contributor experience with nix/nixpkgs. I've mostly used git and GitHub for contributing to college projects, private projects, and some work projects that happen to be public on GitHub.

As a user of nix/NixOS

I've been using NixOS for more than 2 years now. I have also used nix with direnv in some of my personal projects.

Recently, I introduced Nix at my workplace[3], where we now utilize it for cross-compiling a Rust binary for Windows from Linux. Additionally, we've begun incorporating Nix into a couple of our private Rust and python projects.

6 Personal Information

- **Full Name**: Siddhant Kumar
- **Email**: siddhantk232@gmail.com
- **University**: Sant Longowal Institute of Engineering and Technology (`http://sliet.ac.in/`)
- **Location**: Punjab, India
- **Timezone**: IST (GMT+5:30)
- **Website and Blog**: `https://siddhantcodes.netlify.app/`
- **GitHub**: `https://github.com/siddhantk232/`
- **Twitter**: `https://twitter.com/siddhantCodes/`
- **Linkedin**: `https://www.linkedin.com/in/siddhantcodes/`
- **Matrix user link**: `https://matrix.to/#/@siddhant_codes:matrix.org/`

7 References

- Uses of `Path` and other typedefs are gathered using LSP tooling.

- https:
  //github.com/NixOS/nixpkgs/pulls?q=author%3Asiddhantk232
- https://github.com/fastn-stack/fastn/pull/1745