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

Siddhant Kumar

March 18, 2024

1 Motivation

1.1 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++.

1.2 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 403 places[1] in the codebase (including tests). The usage is spread across the codebase with operations like "joining paths"[2] that are Unix specific.
- `PathView`, `Paths` and, `PathSet` also defined in the same header file is referenced 56 times[3] mostly in function parameters.

3 Suggested approach

An approach to carry the porting process is described below. One downside of this approach is that we don't get immediate feedback and we have to carry the whole process in one go until we get rid of all the errors.

3.1 Change `Path` typedefs to use `std::filesystem::path`

Doing this will produce of ton of compilation erros that I will be able to use to drive further changes.

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

This (and the corresponding .cc) file 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.

3.3 Update remaining uses of `Path` in the codebase

With the utilities updated, we will stil have a lot of code that refrences the `Path` typedef. However, many are trivial operations like reading the path and joining two paths or calling utils declared in `libutil/file-system.hh` which we've already ported. All these references will be checked based on order they appear in the compiler error messages.

3.4 Manually check untouched areas where `Path` is used

We might end up with some code that uses `Path` and related typedefs[4] but haven't been touched in the porting process. Some of them should be fine as they're just reading the path but, to make sure, a manual check of all the referencing sites will be performed once we clear all the compiler errors.

4 TODO Timeline

Mention the standard coding period timeline. Then list week-wise work deliverables below

- Initial discussion with the mentor
- Learning Period (can be merged into above?)
- **May 1 - 26** - Community Bonding Period | GSoC contributors get to know mentors, read documentation, get up to speed to begin working on their projects
- **May 27** - Coding officially begins!
- **July 8** Mentors and GSoC contributors can begin submitting midterm evaluations
- **July 12** Midterm evaluation deadline (standard coding period)
- **July 12 - August 19** Work Period | GSoC contributors work on their project with guidance from Mentors
- **August 19 - 26** Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)
- **August 26 - September 2**: Mentors submit final GSoC contributor evaluations (standard coding period)

5 Past experience with nix

5.1 As a contributor

I've made one PR[5] 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 open source

5.2 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[6].

Recently, I introduced Nix at my workplace, where we now utilize it for cross-compiling a Rust binary for Windows from Linux[7]. 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 TODO Conclusion

Don't know if I should conclude. It'd be a nice thing to do for sure.

8 TODO References

- [1] [2] [3] [4] TODO: mention how I got these numbers
- [5] `https://github.com/NixOS/nixpkgs/pulls?q=author%3Asiddhantk232`

- [6] list many projects where I've used nix/flakes
- [7] `https://github.com/fastn-stack/fastn/pull/1745`