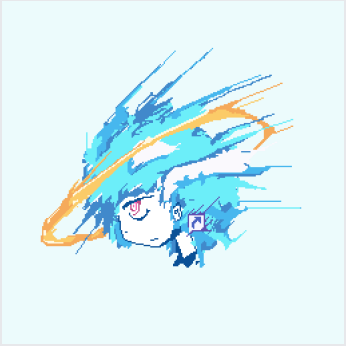
$SAT_{Y}SF_{I}$ w/ Nix

SnO_2WMaN

2022.09.24

はじめに

自己紹介



SnO_2WMaN

興味があるものがあると言ったら嘘になる

- GitHub: @SnO2WMaN
- Twitter: @SnO2WMaN

現状の問題点

SATγSF$_I$の環境構築 (1)

少なくとも私にとっては SATγSF$_I$ での（満足な）執筆環境の構築は未だに結構面倒だった．

- OPAM の操作にクセがありすぎる．
  - 私は OCaml についての講習を受けていません．
- ~~Satyrographos のスペルが覚えづらすぎる．~~
- その他周辺ツールやフォーマッタの用意．

SATγSFIの環境構築 (2)

とはいえ，TEX の環境構築よりは遥かに楽[*1]ではあり，執筆体験的にも TEX に戻るのは少し抵抗がある．
当時の自分は Nix に取り憑かれていたので，なんとか Nix の世界に落とし込めないか？と画策した．~~そんなに文書も書かないのに ...~~

1 Docker 及び devcontainer を使って若干楽に用意する方法がある．

Nix について

Nix とは (1)

関数型パッケージマネージャと呼ばれるもの．大雑把に言えば次のような流れでパッケージを生成する．なお，Nix のパッケージの概念は実際は後述するようにかなり広い範囲を表すため，便宜上，**生成物**と呼ぶ．

1. Derivation を Nix 言語で記述する．
   (a) Derivation はビルド時に上書きできる．
   (b) これらは容易に git で管理出来るという点に注意．
2. Derivation から生成物が生成される．
   (a) この際，依存するツールなどを含めた全ての入力についてのハッシュを計算する．
   (b) このハッシュが一致しているなら同じものが生成される筈なので， その際はキャッシュから返すことでビルドをスキップ出来る．

Nix とは (2)

生成物は実際には `/nix` 以下に全て置かれ，利用者が利用するときにはそれら実体への symlink が上手く張られるようになっている． ところで，`/nix` に置かれるものには以下のものも含まれる．

- 適当なファイル構造
- 開発環境
  - 実際は，プロジェクト内だけで使うツールへの `$PATH` を通したりコマンドを定義したスクリプト
- dotfiles / OS の設定ファイル
  - 例えば `.zshrc` や `.vimrc`，`~/.config/` 以下などを生成，管理[*2]できる．
  - もっと拡張して OS の設定を丸ごと Nix で管理，生成することも出来る．

2 home-manager を参考．

完成したもの

satyxin

https://github.com/SnO2WMaN/satyxin

もともとは AumyF が最初に作ったものを，大幅に魔改造して作った，SATγSF$_{\mathrm{I}}$ 文書を Nix 上でビルドするための様々なものを提供するためのライブラリ．次のことをする．

1. Satyrographos のために書かれた SATγSF$_{\mathrm{I}}$ パッケージを Nix のパッケージとして提供する．
   (a) ただし現状では手動で `Satyrographos` や `.opam` ファイルなどを読んで定義している．
2. `.satysfi/dist` を一種の生成物と考え，生成するための wrapper を提供．
   (a) パッケージから適切なディレクトリにファイルをコピーしたり，フォントのハッシュファイルなどを上手く合体させる．
   (b) これをプロジェクトのルートに置くとローカルでプレビュービルドが出来る．
      (i) 実際には symlink が張られる．
3. `.satysfi/dist` を入力に受け取り，ドキュメントをビルドするための wrapper を提供．

SATYSFI文書を Nix でビルドできることのうれしさ (1)

Nix が入った環境なら**どこでも** 30 秒フラット[*3]で pdf が生成出来る執筆環境をフォーマッタなども含めて構築できる．
例えばこれは，Nix が用意された自分のマシン A で執筆環境を構築できたなら，他の Nix が用意されたマシン B（自分のでも，他人のでも！）でも執筆環境を構築出来る，ということを**保証する** [*4]．これによって，共同執筆の際でも，Nix さえ入れておいてもらえれば楽に環境構築できそう，ということを期待させる．

3 すでにキャッシュされているなら，という条件付きだが．

4 git，VSCode や Vim など，あまりにも大前提すぎるものは通常除く．

SATγSF$_I$文書を Nix でビルドできることのうれしさ (2)

Satyrographos のリポジトリの更新を待たずとも，リポジトリの中だけで使う仮パッケージを作ることも，自由に指定のバージョンや最新版[*5]へ切り替えることが出来る．これは Nix がビルド時に Derivation を上書きできる性質から可能となっている．
例えばこのスライドの中だけでも次のことが行われている．

- figbox パッケージの `feat-include_png` ブランチ．
  - 最新版の SATγSF$_I$ に入った PNG サポートに対応．
- fonts-jetbrains-mono パッケージの仮作成．
  - lmmono フォントがいまいち合っていなかったので．
- SATγSF$_I$ そのもののフォーク時最新の（`master`）ブランチへの追従させている．

5 バージョンである必要もなく，Git のコミット単位で切り替えが可能である．

実証実験

このスライドは，実際に satyxin を使ってビルドされています！

GitHub Actions 上で Nix を用意する Action はすでに用意されており[*6]． 生成物はそのまま GitHub Pages に投げることが出来る[*7]．ビルドの手順は satyxin が隠蔽するため，利用者 /CI 上では `nix build` の 1 行で pdf が生成されるというシンプルさもある．

- リポジトリ
- GitHub Pages 上の pdf ファイル

6 cachix/install-nix-action

7 本題と関係ないが，最近，ベータ版だが `gh-pages` ブランチ無しで Action から直接 GitHub Pages にデプロイが出来るようになっている．

その他の話題

SATγSF$_I$自体を Nix でビルド出来るようにする

Nix の公式パッケージ集で用意されている SATγSF$_I$（現状 v0.0.7） はかなり場当たり的な対応[*8]が行われている．これをちゃんとさせた上で，最新版の PNG サポートが使いたい！という欲求のもと，SATγSF$_I$ 自体を Nix でビルド出来るようにした[*9]．

8 `.opam` で指定されたバージョンを全部無視して nixpkg で提供されている OCaml パッケージでビルドしているなど．

9 苦闘の歴史．

周辺ツール

VSCode で楽しく文書を書くために SATYSFI Formatter や SATYSFI Language Server などもフォークした上で Nix の世界に持ってきている．

- https://github.com/SnO2WMaN/satysfi-formatter/nix-intgl
- https://github.com/SnO2WMaN/satysfi-language-server/nix-intgl

これは本題とは関係ないが，textlint という自然言語に関する linter（二重否定，ら抜き言葉への警告などを行う）も，SATYSFI 上でも使える[*10]ようにし，より文書を書く体験を向上させていきたい．

10 SATYSFI の AST を構成して文章の部分だけ lint をかける為の textlint のプラグインを書く必要がある（はず）．詳しくはよく調べてないが，monaqa/tree-sitter-satysfi とかを使って頑張れば出来るんじゃないか？と思う，誰か挑戦してみてください．

おわりに

課題など

satyxin の現状の課題としてはこのようなものがある．

- 足りてないパッケージについてもドシドシ追加する
- SATγSF$_{\mathrm{I}}$ v0.1.0 で導入が検討されているパッケージ管理システムについても追随できるようにする．
- そもそも Nix のユーザが少なすぎて解説しても旨味がよくわからない．
  - なんとかするために現状 Nix についての本を執筆中です．

satyxin に依らない個人的な課題としてはこのようなものがある．

- 執筆体験のより向上を目指す．
- リポジトリを登録すれば arxiv のような感じで pdf の生成とかも全部やってくれるようなポータルの構築（構想だけ）．
  - 訂正依頼とかは Issue とか Pull Request などで出せば良い．