rustyfi

A native Rust port of SATySFi

目次

1. What this is

`rustyfi` compiles a SATySFi document (`.saty`) to PDF. It is a native Rust port of SATySFi, tracking upstream v0.0.6, with partial support for the 0.1 language (`--target-version 0.1`).

It is not a wrapper around the original: the lexer, parser, elaborator, type checker, evaluator, line breaker, page breaker and PDF writer are all reimplemented. Where behaviour is ported from upstream, the source file and line are cited in a comment at the port site, because the reference implementation is the specification.

2. Compiling a document

The common case needs no flags. Output defaults to the input path with a `.pdf` extension.

```
$ rustyfi document.saty
$ rustyfi document.saty -o out/document.pdf
```

2.1. Finding packages and fonts

`@require:` resolves against a library root — a directory holding `dist/packages/`. The root is taken from `--lib-root`, else `$RUSTYFI_LIB_ROOT`, else the nearest `lib-rustyfi/` found by walking upward from the input file. Fonts are discovered the same way from `--font-dir` (falling back to the library root), or a single face can be passed directly with `--font`.

With no font configured anywhere, text is set in the built-in base-14 faces. That is enough to produce a PDF, but a document containing CJK will need real fonts.

2.2. Output formats

`--format pdf` is the default. `--format html` is a faithful, non-reflowing serialization of the same laid-out pages the PDF writer renders — every run absolutely positioned — intended as a preview and visual-diff aid. `--format html-reflow` is a separate, semantic serialization with real flowing paragraphs and CSS layout: readable, but deliberately not layout-faithful.

3. Incremental builds

Two independent caches make a repeat build cheap, and both can be turned off when measuring.

- The *compile cache* is content-addressed on the source and skips the whole pipeline when nothing has changed. `--no-cache` bypasses it for both reading and writing. Note it is keyed on the SOURCE, not on the compiler binary, so a build that must reflect a change to the engine itself needs this flag.
- The *auxiliary file* `<doc>.satysfi-aux` seeds the cross-reference fixpoint from the previous run, letting a forward reference resolve on the first trial instead of forcing a second. It uses the same name and JSON format as upstream SATySFi, so the two interoperate. `--no-aux` ignores it, and `--aux-file` moves it. Output is identical either way — only the number of trials differs.

`--timing` prints a per-phase breakdown (load, elaborate, typecheck, evaluation trials, render) to standard error. It implies `--no-cache`, so every phase actually runs. It does *not* imply `--no-aux`, because an aux file skips no phase.

4. Packages

The `satyrographos` subcommand is this port's package-manager analog. It installs from a

directory, a `.tar.gz`, or a registry name, and reads either a `rustyfi-package.toml` or an upstream-format `Satyristes`.

```
$ rustyfi satyrographos install ./my-package
$ rustyfi satyrographos list
$ rustyfi satyrographos status
```

A `Satyristes` is read for its `(library ...)` blocks: `(name ...)`, `(version ...)`, `(sources ...)` and `(dependencies ...)`. The `(libraryDoc ...)`, `(opam ...)` and `(compatibility ...)` forms are parsed and ignored, as they describe things outside this port's scope.

5. Fidelity to upstream

Correctness here means agreeing with the original typesetter, so the repository carries a layout-fidelity harness that builds each corpus document with the port and compares the result against a PDF built by the original OCaml SATySFi.

```
$ python3 scripts/layout_fidelity.py \
      --bin "$(pwd)/target/release/rustyfi"
```

Comparison is by word bounding box, not by bytes: the port bundles the same fonts SATySFi uses, so glyph metrics are identical and any divergence reflects the engine — line breaking, inter-box spacing, page breaking, box placement. Two metrics carry most of the signal. `text_match` compares the sequence of words, catching text that is missing, extra or out of order. `width_p95` pairs up words the two engines agree on and reports the 95th percentile of their width difference, which is what exposes a wrong font or a wrong glyph advance.

Page, line and word counts are pinned as a *deviation from upstream* that may shrink but never grow, so a document moving toward SATySFi can never fail the check, and every improvement is locked in.