regexp engine investigation

nix uses libstdc++'s `std::regex`. it uses whatever version of libstdc++ the host system has.

which it invokes in both `std::regex_replace` `std::regex_match` modes.

nix occasionally uses the flags `std::regex::extended` and `std::regex::icase` which determine the available features - it's always either no flags, or both of these together. there's also a couple things that use the flag `std::regex::ECMAScript`. when the constructor is called without a flags parameter, the flags default to `std::regex::ECMAScript` (see method signature in C++23 32.7.2), so really we have only two cases.

`std::cregex_iterator` and `std::sregex_iterator` are used.

there's a header `regex-combinators.hh` which defines `regex::group` and `regex::list`.... and a couple others that are unused. but those are just trivial textual things, not extensions, so we can ignore the file.

getting the C++ standard

someday when C++23 is official you will be able to pirate the PDF. otherwise, you can clone https://github.com/cplusplus/draft and check out the tag `n4950` which is the current formally adopted working draft as of 2024-03-14 and is intended to have the same technical content as the final standard. you can then invoke `make` in the `source` subdirectory which will produce `std.pdf`. you will need LaTeX installed. if you're ever not sure which working draft is the one that became a particular version of the standard, Wikipedia will probably tell you...

(personally I install `texlive.combined.scheme-full` from nixpkgs on all my machines that have room for it, but this is surely more than necessary, it just makes me feel warm and fuzzy -- Irenes)

chapter 32 is the one that documents regular expressions.

open questions that require reading the standard

- what are all the syntactic and semantic constructs we need to support?

required functionality

the `extended` flag, per the C++ standard, "Specifies that the grammar recognized by the regular expression engine shall be that used by extended regular expressions in POSIX.". it references POSIX, Base Definitions and Headers, Section 9.4.

the `ECMAScript` flag "Specifies that the grammar recognized by the regular expression engine shall be that used by ECMAScript in ECMA-262, as modified in [section 32.12 of the C++ standard]." it

references ECMA-262 15.10. the changes in 32.12 are important and probably do create real compatibility issues for us, though fortunately it's only a single page.

if we complete this chart we can use it to assess which existing engines would meet our needs, or how much of a pain in the ass it would be to make a new one

the columns are the two ways it gets invoked

	extended + icase	ECMAScript
Syntactic constructs	--	--
(TODO: fill in every construct here)		
Semantics	--	--
Case-insensitivity	yes	?
(TODO: fill in other behaviors here)		

Revision #1
Created 2024-03-27 07:10:07 UTC by jade
Updated 2026-02-01 13:14:00 UTC by jade