Ferrocene

Qualifying rustc using only open source software
- Florian Gilcher
- Managing Director Ferrous Systems ([https://ferrous-systems.com](https://ferrous-systems.com))
- Previously Rust Foundation, Rust Project core & community team
- Rust Trainer since 2015
- Formed 2018 by Rust project members to help companies adopt Rust
- Mainly a training and maintenance company
- (co-)maintained projects: rust-analyzer, knurling, bindgen, crates.io
- Ferrocene: bringing Rust into safety-critical

- https://ferrous-systems.com
AGENDA SLIDE

What is Ferrocene?

Tools Used

Not Rocket Science

Execution

Experiences
What is Ferrocene?

A qualification of the Rust compiler, rustc

Ferrocene is a complete downstream toolchain of [https://github.com/rust-lang/rust](https://github.com/rust-lang/rust)

ISO 26262 and IEC 61508

A language specification effort

Ferrocene has the only complete specification of Rust 1.68

[https://spec.ferrocene.dev](https://spec.ferrocene.dev)

Long Term Supported Rust

The Rust project supports for 6 weeks, Ferrocene longer

Fully Open Source based

Ferrocene uses open source tools and methodologies derived from the Rust project
Rust Basics: Target Tiers

- Target tiers describe the level of support a target gets
- Tier 1: Fully supported, breakage blocks release, needs full automated testing
- Tier 2: Supported, shipped with the compiler, but may not be automatically tested
- Tier 3: Code is in the repository, but no further guarantees, target is not always shipped
- Example: aarch64-unknown-none is a Tier 2 target upstream, but Tier 1 in Ferrocene
Automatically maintain a repository of code that always passes all the tests.
Not Rocket Science Rule of Software Engineering

Plain CI is not enough for this!
Ethos (Mozilla Research)

What can’t be released, isn’t software.
Every build is a release

The testing of rustc includes building all release artifacts as if a full release is made.
Tools Used

- Sphinx for documentation [https://www.sphinx-doc.org](https://www.sphinx-doc.org)
- Bors-NG for automation [https://bors.tech](https://bors.tech)
- Sigstore for document signing [https://www.sigstore.dev/](https://www.sigstore.dev/)
- Reuse for Licensing/SBOM concerns [https://reuse.software/](https://reuse.software/)

- GitHub as a Platform [https://github.com](https://github.com)
- CircleCI as a Build Platform [https://circleci.com](https://circleci.com)
BORS-ng

- Maintains a queue of PRs that are ready to be merged (reviewed)
- Serializes them for testing, so only one PR runs at a time
- Automates all operations on the repository
- Does all important bookkeeping, e.g. tracking who reviewed and who authored a change
- Signals back on failures
BORS-ng

pietroalbini approved these changes last week

pietroalbini left a comment

bors merge

bors-ferrocene: bot commented last week

- Waiting for PR status (Github check) to be set, probably by CI. Bors will automatically try to run when all required PR statuses are set.

bors-ferrocene: bot commented last week

Build succeeded:
- full

bors-ferrocene: bot merged commit 6e19228 into release/1.68 last week

3 checks passed

bors-ferrocene: bot deleted the automation/backport/w28olbu branch last week

View reviewed changes
BORS-ng: Merge queue

Feature branch merge tests

<table>
<thead>
<tr>
<th>feature/x</th>
<th>feature/x</th>
<th>main  (feature/x+feature/y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature/y</td>
<td>feature/y</td>
<td></td>
</tr>
</tbody>
</table>

Feature branches with merge queue

<table>
<thead>
<tr>
<th>feature/x</th>
<th>feature/x</th>
<th>main  (feature/x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature/y</td>
<td>feature/x + feature/y</td>
<td></td>
</tr>
</tbody>
</table>
Merge #1306 #1307

1306: Allow omitting new-branch argument in generate_pr_body.py  
```plain
r=pietroalbini a=Veykril
```
Makes manual pulls slightly more convenient

1307: Use default implementation for compiletest's config  
```plain
r=Veykril a=pietroalbini
```
This should avoid merge conflicts when new compiletest options are added. The `Default` impl was added in rust-lang/rust#111348.

Co-authored-by: Lukas Wirth <lukas.wirth@ferrous-systems.com>
Co-authored-by: Pietro Albini <pietro.albini@ferrous-systems.com>
More Testing!

Ferrocene considers the qualification material and tracing part of the software test. It is never allowed to break.
## 7.1. Constants

**Syntax**

```
ConstantDeclaration ::= const (Name | _) TypeAscription ConstantInitializer? ;
```

```
ConstantInitializer ::= = Expression
```

**Legality Rules**

1. A constant is an immutable value whose uses are substituted by the value.
2. An unnamed constant is a constant declared with character 0x5F (low line).
3. The type specification of a constant shall have "static" lifetime.
### Traceability Matrix

<table>
<thead>
<tr>
<th>Fls: Values</th>
<th>7.1 Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="fls_ixjc5jaam84" alt="image" /></td>
<td><img src="tests/ui/const-generics/generic_arg_infer/in-signature.rs" alt="show 690 linked tests" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/array-literal-index-oob.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/array-literal-len-mismatch.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/array-to-slice-cast.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/assert-type-intrinsic.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/assoc-const.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/assoc-const-generic_impd.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/associated-const_generic.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/async-block.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/bawap-const.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/cast-discriminant-variant.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/chained-constants-stackoverflow.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/check_const-feature-gated.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/closure-in-foreign-crate.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/closure-structural-match-issue-90013.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/const-address-of-interior-mut.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/const-address-of-mut.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/const-address-of-of.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/const-adt-alias-mismatch.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/const-array-cof-arith.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/const-array-cof.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
<tr>
<td><img src="tests/ui/constgs/const-as-fn.rs" alt="image" /> (annotated in its parent directory)</td>
<td><img src="tests/ui/constgs/const-as-fn.rs" alt="image" /> (annotated in its parent directory)</td>
</tr>
</tbody>
</table>
No Such Thing As A Partial Build
## Platform Automation

<table>
<thead>
<tr>
<th>Issue</th>
<th>Title</th>
<th>Status</th>
<th>Assignee</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1321</td>
<td>Backport to release/1.71</td>
<td>Open</td>
<td>automation, backportnever</td>
<td>Github-actions bot opened 1 hour ago by github-actions bot</td>
</tr>
<tr>
<td>#1318</td>
<td>Manual backports into 1.68</td>
<td>Closed</td>
<td>backportnever</td>
<td>Opened yesterday by pietroalbini</td>
</tr>
<tr>
<td>#1317</td>
<td>Pull upstream 2023 07 04</td>
<td>Approved</td>
<td></td>
<td>Opened yesterday by Veykri</td>
</tr>
<tr>
<td>#1316</td>
<td>Backport to release/1.70</td>
<td>X</td>
<td>automation, backportnever</td>
<td>Github-actions bot opened yesterday by github-actions bot</td>
</tr>
<tr>
<td>#1315</td>
<td>Backport to release/1.68</td>
<td>X</td>
<td>automation, backportnever</td>
<td>Github-actions bot opened yesterday by github-actions bot, Approved</td>
</tr>
<tr>
<td>#1312</td>
<td>Automated pull from ferrocene/specification</td>
<td>X</td>
<td>automation, backportnever</td>
<td>Github-actions bot opened yesterday by github-actions bot, Approved</td>
</tr>
</tbody>
</table>
Experiences

- Major FOSS projects already *informally* do what safety-critical projects do *formally*

- When downstreaming a project, it is very useful to play along with their rules, as much as one wants to redo them

- There’s many useful FOSS projects that are very useful, mature and accepted in an enterprise setting

- The ability to tune software to our needs is core to our velocity

- Simple rules without branches and conditions are *possible* and *easy to automate*

- Automating and using our platforms API is worth all the time spent

- *There is no such thing as setting up automation too early*
THANK YOU!