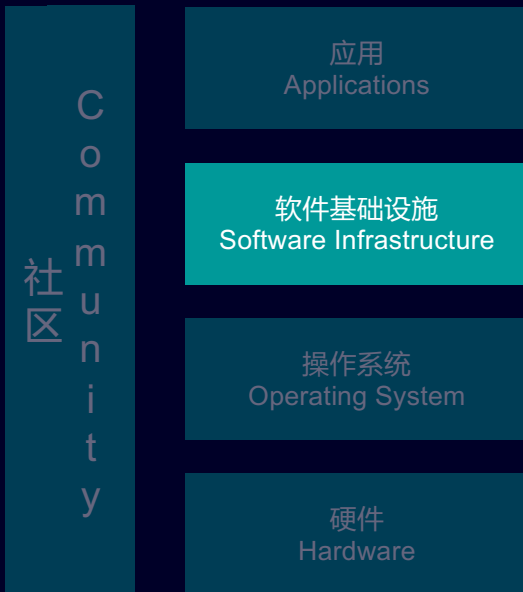


Rust - Open Source Safety



演讲人 Speaker: Florian Gilcher (Ferrous Systems)

主题 Title: Rust开源功能安全 Rust - Open Source Safety

内容 Description

Ferrocene是一个经认证的开源Rust编译器工具链。通过它，Rust已成为一流的针对关键任务和功能安全系统的语言。

Ferrocene is an open source qualified Rust compiler toolchain. With this, Rust has become a first-class language for mission-critical and functional safety systems.

Rust at Siemens

- Internal Rust User Group with ~100 members [**JOIN US**]
- many, many underground playgrounds
- and first products...

Rust use-cases at Siemens Mobility Rail Infrastructure

- Security Infrastructure for Onboard Train Control Network
- Software Components for non-vital communication equipment
- ...



Rust use-cases at Siemens Mobility Rail Infrastructure

- Security Infrastructure for Onboard Train Control Network
- Software Components for **non-vital** communication equipment
- ...



The Sealed Rust Pitch

Sealed Rust

Sealed Rust is Ferrous Systems' effort to **qualify** the Rust Programming Language for **Mission Critical** and **Safety Critical** software development

Statement by Florian in mid 2020

Ferrocene: Enabling Rust in Critical Settings

While keeping it all in the open...

Ferrous Systems (est. 2018)



Ferrous Systems is an open source maintainer company. With a team of 20 awesome people.

Open Source

rust-analyzer

Knurling

Bindgen

sudo-rs

<https://ferrous-systems.com/open-source>



Training

Rust introduction and
advanced

Team augmentation

Management training

Ferrocene trainings

<https://ferrous-systems.com/training>



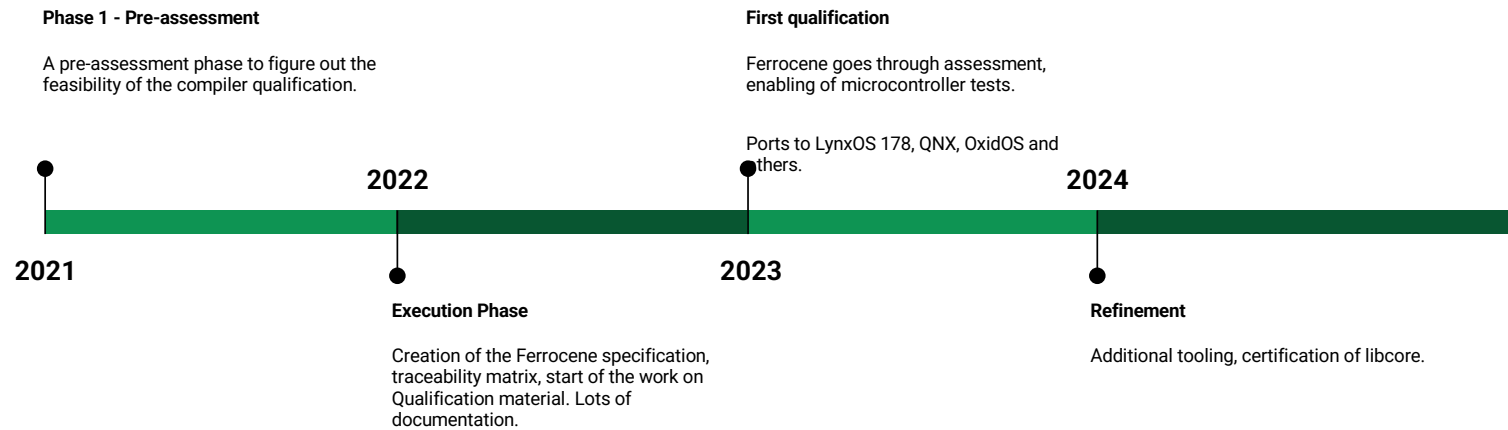
Custom development

Custom development of libraries,
tools and board support
packages.

<https://ferrous-systems.com/contact>



The hidden project: Ferrocene



What do we need for the qualification?



- The tool itself
 - The compiler
- With a covering requirements list
 - A specification
- A set of evidence that shows that the requirements are met
 - Tests, mapped to the specification
- Evidence of development processes that keep these requirements met

Goals of Ferrocene



- Explore ways to create safety evidence for Rust
- Produce a reusable compiler that fits that end
- Be a downstream to rust-lang/rust
- Apply the “Not rocket science rule of software quality” to qualification

Automatically maintain a repository of code that always passes all the tests.

No detour: Functionality

Not a subset

Ferrocene allows usage of the *whole* language surface of the Rust programming language

Paranoid mode for Rust

Ferrocene is configured in **paranoid mode**, which makes it a bit slower than the Rust compiler, but also catches more bugs.

All versions available

Ferrocene ships all future versions of the Rust compiler, with out-of-band qualification available on request.

Findings



High quality of rustc

The Rust compiler is tested to a very high degree already



Great process landscape

The Rust Project's processes are quite fitting for a high assurance mindset



No need for divergence

Ferrocene does *not* diverge from the main Rust compiler and its test suite, only adds things around it.

Current status



As of October 30th, Ferrocene has a certificate by TÜV SÜD.

It can be used in ISO 26262 (automotive) and IEC 61508 (industry) settings in all criticality levels on ARMv8-A platforms.

Currently missing

A bit of polish

- We're finishing our install story!
- We still have a waitlist of customers to work through
 - Currently serving organisations of 10 and up
- A faster development version of the Ferrocene compiler – paranoia costs compile time
- Enablement of many nearly-finished targets

Ferrocene's Model: Affordable Open Source for High Assurances

Affordable base package

Access to Ferrocene binaries is 25 EUR/month, 240 EUR/year

Those binaries come with access to extended support periods.

All documentation included.

No license locking.

DevOps ready – we won't charge for CI use.

Extended Support

Ferrocene shipped compilers are supported longer than the Rust projects timeframe.

Development support is also available.

Active Notification

We actively inform you of issues of the tools you are currently using along with fixes and appropriate mitigations.

Ferrocene's Model: Be as flexible as any FOSS project

Functional Safety

Certification support to make your system certification go smoothly. This includes access to signed versions of all material.

LTS

Long Term Support beyond the usual timeframes.

Devops Use

You want to ship, deploy or even build Ferrocene through your own infrastructure? We're happy to help!

SDK integration

Ferrocene considers custom, vendor-managed toolchains first class.

Air-Gapping

Solutions for Air-Gapped environments will be available soon.

Custom Development

Need additions or something that currently doesn't exist? Let's talk.

Open Source

Ferrocene's Model: Sustainable Open Source for High Assurances

Open Source Code

Ferrocene's tool code is open source under Apache-2.0/MIT.

<https://github.com/ferrocene/ferrocene>

Opens Source Docs

User, Safety and Qualification manuals and all evidences are available as Apache-2.0/MIT.

<https://public-docs.ferrocene.dev/main/>

Public Transparency

We commit to updates similar to our other projects.

<https://ferrous-systems.com/blog>

Ferrocene does not accept open contribution, but we are currently designing easy contribution paths. The main blocker here is the requirements of the quality management system. Compiler fixes should go to the main Rust project.

We're happy to take feedback!

Why Open Source?

There is currently a new move to open source in vehicle industries

<https://www.soafec.io/> - Open Source Architecture initiative

<https://covesa.global/> - Connected Vehicle Systems Alliance

<https://sdv.eclipse.org/> - Eclipse Software Defined Vehicle

There's frequently issues around required non-open-source tooling

Project contributions

Added structured output for the build system ([#93717](#), [#108659](#), [#111936](#))

Pinned the checksums of the bootstrap compiler ([#88362](#))

Made it easier to use a custom libc ([libc#3037](#), [#108898](#))

Validated ignore annotation in tests ([#108905](#), [#110319](#))

Started clarification of licensing ([#99415](#), [#104139](#), [#104439](#), [#104527](#), TBC)

Fixed tests on panic=abort [bare-metal] targets ([#111992](#), [#112314](#), [#112418](#), [#112454](#))

Fixed tests inside long filesystem paths ([#96551](#), [#97786](#))

...plus lots of small miscellaneous fixes and changes!

Fully Open Source tool stack

Reuse

Tool produced the FSFE to improve license checking.

<https://reuse.software>

Sphinx

The trusted workhorse of documentation tooling.

<https://www.sphinx-doc.org>

Sigstore

We digitally sign documents in our CI pipeline.

<https://www.sigstore.dev/>

bors-NG

Process automation bot. What's a Rust project without bors?

<https://bors.tech>

cargo-dist

Axo powers our release story.

<https://opensource.axo.dev/cargo-dist/>

Python & Bash

Is there software without it?

Outcome: lots of additions



Showing 5,084 changed files with 69,823 additions and 161 deletions. Split Unified

Filter changed files

- └─ .circleci
 - └─ README.md (+)
 - └─ calculate-parameters.py (+)
 - └─ config.yml (+)
 - └─ workflows.yml (+)
- └─ .gitattributes (🔍)
- └─ .github/workflows
 - └─ automation-backport.yml (+)
 - └─ automation-pull-subtrees.yml (+)
 - └─ automation-pull-upstream.y... (+)

The diff you're trying to view is too large. We only load the first 3000 changed files.

26 █████ .circleci/README.md

```
... @@ -0,0 +1,26 @@  
1 + <!-- SPDX-License-Identifier: MIT OR  
  Apache-2.0 -->  
2 + <!-- SPDX-FileCopyrightText: The  
  Ferrocene Developers -->  
3 +  
4 + # CircleCI configuration  
5 +  
6 + This directory contains the  
  configuration for Ferrocene's CI,  
  powered by  
7 + CircleCI.
```

And a nice detour: <https://spec.ferrocene.dev>



ferrocene
Language Specification

Search...

Contents:

1. General
2. Lexical Elements
3. Items
4. Types and Traits
5. Patterns
6. Expressions
7. Values
8. Statements
9. Functions
10. Associated Items
11. Implementations
12. Generics
13. Attributes
14. Entities and Resolution
15. Ownership and Destruction
16. Exceptions and Errors
17. Concurrency
18. Program Structure and Compilation
19. Unsafety

Ferrocene Language Specification

Contents:

- 1. General
 - 1.1. Scope
 - 1.2. Versioning
 - 1.3. Definitions
- 2. Lexical Elements
 - 2.1. Character Set
 - 2.2. Lexical Elements, Separators, and Punctuation
 - 2.3. Identifiers
 - 2.4. Literals
 - 2.5. Comments
 - 2.6. Keywords
- 3. Items
- 4. Types and Traits
 - 4.1. Types
 - 4.2. Type Classification
 - 4.3. Scalar Types
 - 4.4. Sequence Types
 - 4.5. Abstract Data Types
 - 4.6. Function Types
 - 4.7. Indirection Types
 - 4.8. Trait Types
 - 4.9. Other Types
 - 4.10. Type Aliases

Documentation



Ferrocene documentation

User Manual

An easy-to-read, how-to manual for operating Ferrocene in a qualified environment.

Language specification

The expected behavior of the Rust language as implemented by the Ferrocene compiler.

Safety Manual

Usage and constraints information concerning the use of Ferrocene according to the functional safety standards.

Safety Manual (excerpt)

11. Tool Options



Ferrocene is qualified exclusively for the following command line options:

- Users shall pass command line option `--edition 2021` to each invocation of the Ferrocene compiler.
- Users shall pass command line option `-C opt-level=2` to each invocation of the Ferrocene compiler.
- Users shall pass command line option `-C llvm-args=-protect-from-escaped-allocas=true` to each invocation of the Ferrocene compiler.
- Users shall pass all target-specific command line options, as listed in the page of the target in the [Compilation Targets](#) section of the User Manual.

Plan & Report



Qualification Material

Evaluation Plan



Evaluation of applicable requirements to determine Ferrocene compiler qualification levels and activities to be performed.

Evaluation Report

Evaluation of all hazardous events that can happen on the compiler and setting counter-measures in order to mitigate these hazards.

Qualification Plan




Phases and activities that describe the qualification of the Ferrocene compiler with the TCL 3/ASIL D level in accordance with ISO-26262 standard and the T3 TQL level in accordance with the IEC-61508 standard.


Qualification Report

How Ferrocene is qualified, including which tests can be used to prove functionality and validation goals.

<https://public-docs.ferrocene.dev/>

How we use the spec


Language Specification

Search... 

Contents:

- 1. General
- 2. Lexical Elements
- 3. Items
- 4. Types and Traits
- 5. Patterns
- 6. Expressions
- 7. Values
- 8. Statements
- 9. Functions

7.1. Constants

Syntax

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

Legality Rules

- 7.1.1 A **constant** is an **immutable value** whose uses are substituted by the **value**.
- 7.1.2 An **unnamed constant** is a **constant** declared with character 0x5F (low line).
- 7.1.3 The **type specification** of a **constant** shall have `'static lifetime`.

How we use the spec: Tracing

FLS: Values 7.1 Constants fls_ixjc5jaamx84 ▼ show 690 linked tests

Requirements

Test mapping is a good task to start early, as it is repetitive and boring. It's better done a few hours at a time.

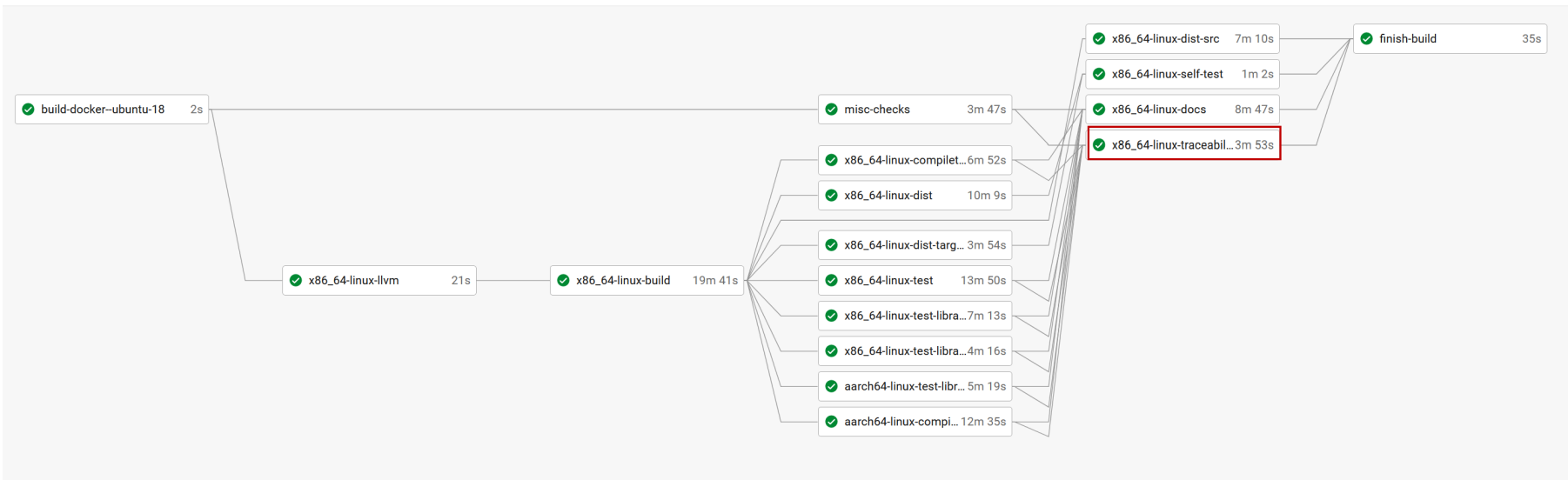
Tests

- [tests/ui/const-generics/generic_arg_infer/in-signature.rs](#)
- [tests/ui/consts/array-literal-index-oob.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/array-literal-len-mismatch.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/array-to-slice-cast.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/assert-type-intrinsics.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/assoc-const.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/assoc_const_generic_impl.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/associated_const_generic.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/async-block.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/bswap-const.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/cast-discriminant-zst-enum.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/chained-constants-stackoverflow.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/check_const-feature-gated.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/closure-in-foreign-crate.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/closure-structural-match-issue-90013.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-address-of-interior-mut.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-address-of-mut.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-address-of.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-adt-align-mismatch.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-array-oob-arith.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-array-oob.rs](#) (annotated in its [parent directory](#))
- [tests/ui/consts/const-as-fn.rs](#) (annotated in its [parent directory](#))

No such thing as a partial build

Dashboard Project Branch Workflow
All Pipelines > ferrocene > staging > full
full Success

Duration / Finished 43m 48s / 15h ago
Branch staging
Commit f6f6fa2
Author & Message Merge #1320



Ferrocene's Model: Constant testing and merging of upstream

The screenshot shows the GitHub interface for the repository `ferrocene / ferrocene`. The navigation bar includes links for Code, Issues (14), Pull requests (5), Discussions, Actions, Security (9), Insights, and Settings. The search bar contains the query `is:pr is:closed Automated pull`. Below the search bar, there are filters for Labels (4) and Milestones (0), and a green button for "New pull request".

The main content area displays a list of pull requests with the following details:

Open	Closed	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from <code>rust-lang/libc</code> ✓ automation backport:never #56 by github-actions[bot] was merged 3 days ago • Approved								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from <code>ferrocene/sphinx-shared-resources</code> ✓ automation backport:never #55 by github-actions[bot] was merged 3 days ago • Approved								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from <code>rust-lang/libc</code> ✓ automation backport:never #52 by github-actions[bot] was merged 4 days ago • Approved								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from upstream <code>master</code> ✓ automation backport:never #50 by github-actions[bot] was merged last week • Approved								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from upstream <code>beta</code> ✓ automation backport:never #49 by github-actions[bot] was merged last week • Approved								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Automation	automation	backport:never				
Automated pull from <code>rust-lang/libc</code> ✓ automation backport:never #47 by github-actions[bot] was merged last week • Approved								

BORS-ng

- Maintains a queue of PRs that are ready to be merged (reviewed)
- Serializes them for testing, so only one change 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 status changes (success/failures)

BORS-ng: your friendly assistant for repetitive workflows

The screenshot displays a vertical timeline of commit history for a repository. At the top, a user named 'pietroalbini' is shown with a green checkmark icon, indicating an approval. Below this, a comment from 'pietroalbini' is shown, containing the text 'bors merge' and a smiley face emoji. This is followed by a comment from the 'bors-ferrocene' bot, which states: '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.' Below that, another comment from the 'bors-ferrocene' bot reports a successful build: 'Build succeeded:' followed by a bulleted list containing 'full'. The next entry shows the 'bors-ferrocene' bot merging commit '6e19a28' into the 'release/1.68' branch, with a note that '3 checks passed'. At the bottom, the 'bors-ferrocene' bot is shown deleting the 'automation/backport/w28o1bbu' branch.

pietroalbini approved these changes last week [View reviewed changes](#)

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 6e19a28 into release/1.68 last week [View details](#) [Revert](#)
3 checks passed

bors-ferrocene (bot) deleted the automation/backport/w28o1bbu branch last week [Restore branch](#)

BORS-ng: forced merges

Merge #1306 #1307

1306: Allow omitting new-branch argument in generate_pr_body.py r=pietroalbin a=Veykril

Makes manual pulls slightly more convenient

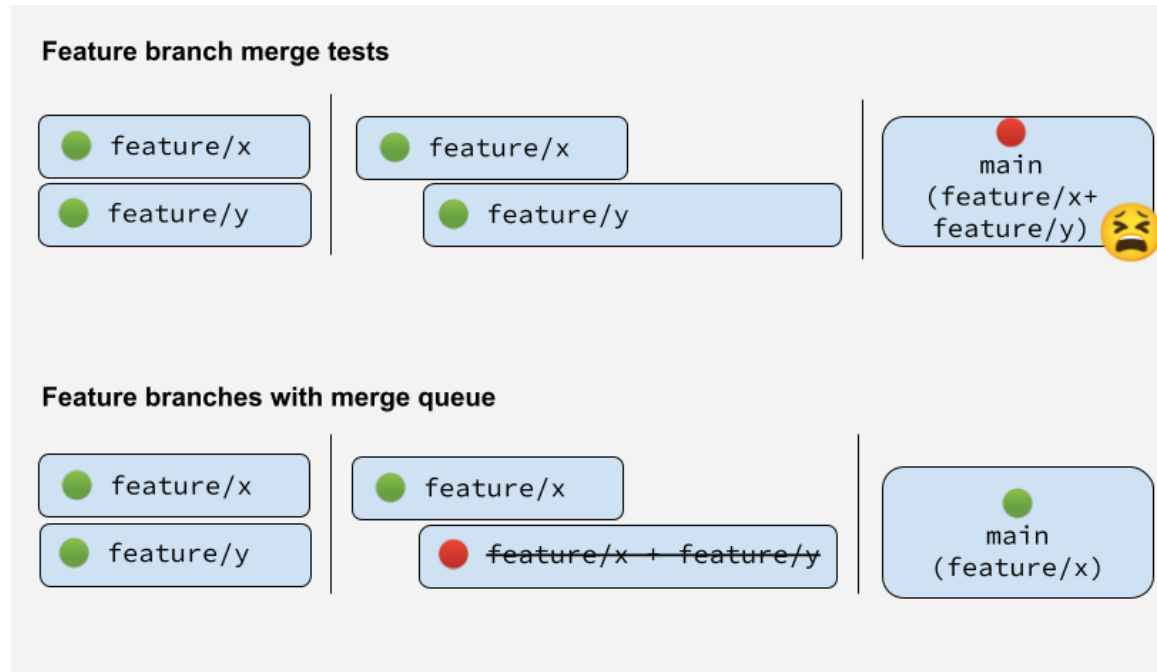
1307: Use default implementation for completest's config r=Veykril a=pietroalbin

This should avoid merge conflicts when new completest 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 Albin <pietro.albin@ferrous-systems.com>

Bors-ng



bors-ng maintains a constantly running list of future builds.

Goal: Straight-Forward Path to Improvement

More and improved testing!

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

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 upstream 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!

Need assistance with your next project?
Get in touch with us.