“how is software safety certified?” (a crash course)

I get asked about this pretty often

especially with "safety critical rust compilers"

- ferrocene
- adacore
- hitec

I used to work in safety critical

(avionics, gas detection, industrial)

this is a:

- crash course
- faq
- common falsehoods

not enough info to ship your first device :)

there is no "safe"
and "not safe" binary

no one-size fits all approach

you can't just make a "safe component"

what is a "safe bolt"?

functional safety

IMO a few key ideas:

failure is *statistical*

everything can fail.

failure has a *price*

deaths, injuries,
financial,
in that order.

we must reason about failure as an engineering problem

address risk

the best way we know

how to

this approach is tailored industry to industry

functional safety is VERY process oriented

defined ways to:

say what you are going to do (specification)

do what you say you are going to do (implementation)

prove you did what you said you would do (verification)

the point:

a paper trail

a chance to catch issues BEFORE they become a failure

everything can fail,
especially people.

only "end products"
are safety certified

context matters!

tools and libraries are components

you can develop components "up to a standard"

this makes it easier for people making the "end product"

"just add safety"
afterwards is hard
(often impossible)

working backwards to prove safety is often harder than rewriting

the Rust Compiler is a rare exception!

LOTS of diligence was done along the way:

docs, decisions, reviews, testing, in writing, with history!

Rust did this because it was important to get right!

this formalism helped make a safety case too!

why does safety critical "move slow"?

we can mitigate known failure modes

see: MISRA

we CAN'T mitigate "unknown" failure modes

risk/reward benefit has to be considered!

the wheels move, just slowly.

disclaimer: this is all "intent" of functional safety

think of it like "the scientific method"

if you just check a box, it's less valuable

functional safety is imperfect, but useful.