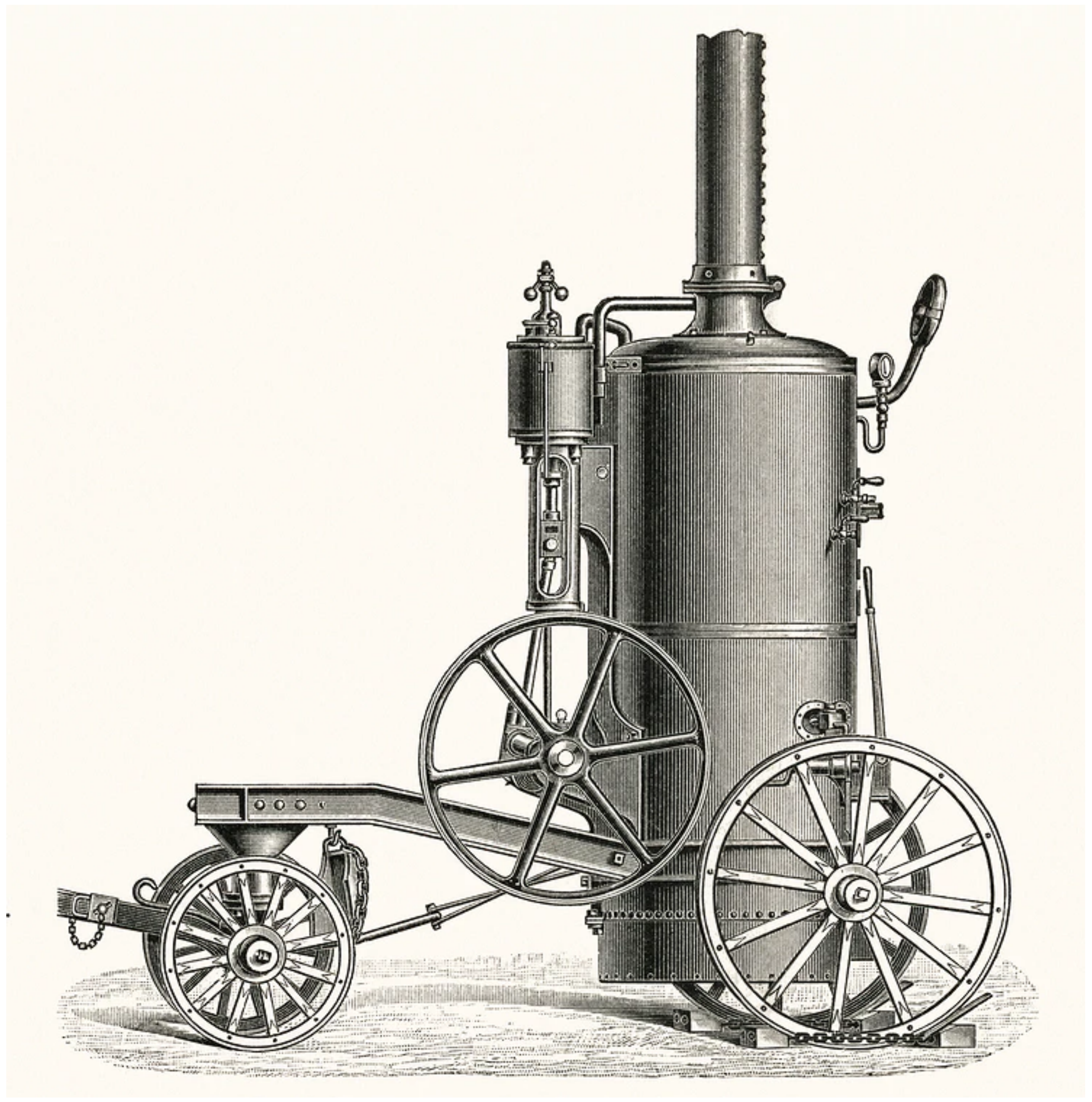

Rust by example: asn-db and asn-tools

Rust features by example with asn-db and asn-tools crates.

Code presentation

In this presentation, I will go through the source code of `asn-db` and `asn-tools` crates. I will explain their structure and basic Rust features according to the plan outlined here. This will be a 10000-foot view on programming in Rust.

I hope you can all follow along.
Please stop me and ask questions at any time.

Autonomous Systems number database

- What is ASN
- IPtoASN website and database files
- `asn-db` and `asn-tools`
- `asn-lookup` command demo
- Post about the presented crates: https://jpastuszek.net/asn/

Rust crates

- Library vs binary crate
- Binary crates with library code
- Using libraries with `cargo add` (`cargo install cargo-edit`)

Compiling and running

- `cargo check`, `cargo build`, `cargo install` and `cargo run`
- Running tests with `cargo test`

Crates documentation

- Module level
- Item level
- Comments

Imports

- Modules in Rust
- Visibility
- Re-exports

Global variables

- `const` expressions, "life before main", `lazy_static!` and ongoing work on const generics
- Literal string vs literal binary string
- UTF-8 encoding of source files

Type in Rust

- Primitive types
- `structs`, tuples and named tuples
- `enum` sum type
- Functions and closures are first class objects

Functions, methods and traits

- Free functions
- Methods and functions
- Implementing traits and default implementations

Generics

- Generics
- Trait bounds
- impl Trait syntax
- Lifetime parametrisation

Deriving trait implementations

- Deriving Debug and Clone trait
- Manual implementation of PartialEq and Eq derive
- Deriving Serialize with serde procedural macros

Implementing custom error types

- `Result` type
- `Error` trait and implementation
- `From` trait and implementation
- `?` operator and de-sugaring

Iterators

- Option type
- .next()
- Iterator composability and „zero-cost abstraction”
- Creating iterators with IntoIterator trait
- Collectig iterators with collect/FromIterator and "turbofish" type annotations

CSV parsing

- Builder pattern
- `Read` trait and I/O in rust

Serialization

- `Write` trait
- Writing and reading data with `serde` crate

Panics

- Aborting and unwinding
- Explicit `panic!()` and implicit panic with `[]` index operator
- Panic safety of libraries

Testing

- Writing unit tests
- Running tests
- Note on parallel execution

Command-line applications

- Multiple binaries
- `main` function and its return type
- Accessing arguments and environment

Parsing command-line arguments with StructOpt

- StructOpt and clap crates
- Annotating arguments
- Parsing custom argument types with FromStr
- Accessing values
- Help message

Input and output

- Printing to `stdout` and `stderr`
- Logging with `log` crate
- Reading from `stdin`

Dynamic types

- References, `Sized` trait and stack
- Heap allocations and `Box` reference
- Trait objects and `dyn`
- `vtable` and „zero-cost abstraction”

Thank you!

Q & A