What Is Rust Doing Behind the Curtains?
Hi! I am
Matthias Endler
You might know me from...

My YouTube channel!
My Blog!
Not at all!
- Hotel Search Platform
- 2.5m+ Hotels/Accommodations
- IT departments in Düsseldorf, Leipzig, Palma, Amsterdam
- Java, Kotlin, Go, PHP, Python (, Rust?)
- tech.trivago.com
Why should I care?
Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety. It aims to bring modern language design and an advanced type system to systems programming. Rust does not use a garbage collector, using advanced static analysis to provide deterministic drops instead.
Rust is a systems programming language that runs blazingly fast, prevents *segfaults*, and guarantees *thread safety*. It aims to bring modern language design and an *advanced type system* to systems programming. Rust does not use a *garbage collector*, using advanced *static analysis* to provide *deterministic drops* instead.
Empowering everyone to build reliable and efficient software.
Be curious. Try crazy things. Don't be afraid.
Being curious is an amazing trait! We should embrace it, and help people be curious.

Pascal Hertleif – Rust’s approach of getting things right
Julia Evans

SO YOU WANT TO BE A WIZARD by Julia Evans

What is using all of my CPU!!
Let's ask perf!

Profiling & Tracing WITH PERF by Julia Evans

Here's how I approach learning hard things and getting better at programming!

perf lets you
Profile your Programs!
Trace system calls with low overhead!
And more!

Let's Learn tcpdump by Julia Evans

What are those computers saying to each other?
With tcpdump, you can find out!
Why's Poignant Guide to Ruby

Chunky bacon!!

Chunky bacon!!
The Rust Compiler
RUST SOURCE
HIR
MIR
LLVM IR
MACHINE CODE

Parsing
Desugaring
Borrow-checking
Optimization
Optimization
Optimization
Desugaring...

Candy designed by Freepik, Vegetables by Macrovector
At last...

Code examples!
Example 1
fn main() {}
extern crate std;
use std::prelude::v1::*;

fn main() {}
Types:  
\begin{itemize}
  \item `std::boxed::Box`
  \item `std::option::Option::{self, Some, None}`
  \item `std::result::Result::{self, Ok, Err}`
  \item `std::string::String;`
  \item `std::vec::Vec`
\end{itemize}

Traits:  
\begin{itemize}
  \item `std::borrow::ToOwned`
  \item `std::clone::Clone`
  \item `std::cmp::{PartialEq, PartialOrd, Eq, Ord}`
  \item `std::convert::{AsRef, AsMut, Into, From}`
  \item `std::default::Default`
  \item `std::iter::{DoubleEndedIterator, ExactSizeIterator}`
  \item `std::iter::{Iterator, Extend, IntoIterator}`
  \item `std::marker::{Copy, Send, Sized, Sync}`
  \item `std::ops::{Drop, Fn, FnMut, FnOnce}`
  \item `std::slice::SliceConcatExt`
  \item `std::string::ToString`
\end{itemize}

Functions:  
\begin{itemize}
  \item `std::mem::drop`
\end{itemize}
Types: `Box`, `Option`, `Result`, `String`, `Vec`

Traits: `PartialEq`, `PartialOrd`, `Eq`, `Ord`, `AsRef`, `AsMut`, `Into`, `From`, `ToOwned`, `Clone`, `ToString`

- **Ordering things**
- **Converting things**
- **Default values**
- **Iteration**
- **Marker traits**
- **Calling/Dropping objects**
- **Concatenate objects** (like strings or vectors)

- `DoubleEndedIterator`, `ExactSizeIterator`
- `Iterator`, `Extend`, `IntoIterator`
- `Copy`, `Send`, `Sized`, `Sync`
- `Drop`, `Fn`, `FnMut`, `FnOnce`
- `SliceConcatExt`
Example 2

Ranges
for i in 0..3 {
    // do something with i
}

let range = 0..3;
for i in range {
    // do something with i
}
let range = Range {0, 3};
for i in range {
    // do something with i
}
let range = Range {0, 3};
for i in range {
    // do something with i
}
use std::ops::Range;

let range = Range { start: 0, end: 3 };

for i in range {
    // do something with i
}

use std::iter::IntoIterator;
use std::ops::Range;

let range = Range { start: 0, end: 3 };
let mut iter = IntoIterator::into_iter(range);

while let Some(i) = iter.next() {
    // do something with i
}
use std::iter::IntoIterator;
use std::ops::Range;

let range = Range { start: 0, end: 3 };
let mut iter = IntoIterator::into_iter(range);

loop {
    match iter.next() {
        Some(i) => { /* use i */ },
        None => break,
    }
}
cargo inspect
cargo-install cargo-inspect

cargo inspect foo.rs
Example 3
Ranges - Part II
for i in 0..=3 {
    // do something with i
}
use std::iter::IntoIterator;
use std::ops::RangeInclusive;

let range = RangeInclusive::new(0, 3);
let mut iter = IntoIterator::into_iter(range);

loop {
    match iter.next() {
        Some(i) => { /* use i */ },
        None => break,
    }
}
cargo inspect --diff foo.rs,bar.rs
Example 4

Opening Files
use std::fs::File;
use std::io::Error;

fn main() -> Result<(), Error> {
    let f = File::open("file.txt")?;
    Ok(())
}
use std::fs::File;
use std::io::Error;

fn main() -> Result<(), Error> {
    let f = match File::open("file.txt") {
        Ok(file) => file,
        Err(err) => return Err(err),
    };
    Ok(())
}
use std::fs::File;
use std::io::Error;
use std::convert::From;

fn main() -> Result<(), Error> {
    let f = match File::open("file.txt") {
        Ok(file) => file,
        Err(err) => return Err(From::from(err)),
    };
    Ok(())
}
fn main() {  
    let mut vec = vec![1, 5, 10, 2, 15];  
    vec.sort();
}
More cargo tools!

- cargo-expand
- cargo-asm
- cargo-bloat
Rust Playground
```rust
pub fn square(num: i32) -> i32 {
    num * num
}
```

```rust
bb0: {
    StorageLive(_2); // bb0[0]: scope 0 at src/lib.rs:2:5: 2:8
    _2 = _1; // bb0[1]: scope 0 at src/lib.rs:2:5: 2:8
    StorageLive(_3); // bb0[2]: scope 0 at src/lib.rs:2:11: 2:14
    _3 = _1; // bb0[3]: scope 0 at src/lib.rs:2:11: 2:14
    _4 = CheckedMul(move _2, move _3); // bb0[4]: scope 0 at src/lib.rs:2:5: 2:14
    assert(!move (_4: bool), "attempt to multiply with overflow") -> bb1; // bb0[5]: scope 0 at src/lib.rs:2:5: 2:14
}

bb1: {
    _0 = move (_4: i32); // bb1[0]: scope 0 at src/lib.rs:2:5: 2:14
    StorageDead(_3); // bb1[1]: scope 0 at src/lib.rs:2:13: 2:14
    StorageDead(_2); // bb1[2]: scope 0 at src/lib.rs:2:13: 2:14
    return; // bb1[3]: scope 0 at src/lib.rs:3:2: 3:2
}
```
Compiler Explorer
```rust
pub fn square(num: i32) -> i32 {
    num * num
}
```

```assembly
push    rax
imul    edi, edi
seto    al
test    al, 1
mov     dword ptr [rsp + 4], edi
jne     .LBB0_2
mov     eax, dword ptr [rsp + 4]
pop     rcx
ret
.LBB0_2:
lea    rdi, [rip + .L__unnamed_1]
mov    rax, qword ptr [rip + core::panicking::p
call    rax
ud2
str.0:
.ascii  "/tmp/compiler-explorer-compiler181114-
str.1:
.ascii  "attempt to multiply with overflow"
.L__unnamed_1:
.quad str.1
.quad 33
.quad str.0
.quad 65
.long 3
.long 5
```

godbolt.org
Lessons Learned

- Rust allows for lots of syntactic sugar
- It's good to be reminded about that sometimes
- Tools help us understand what's going on behind the curtains.
Now go and build cool things!
<table>
<thead>
<tr>
<th>Issue</th>
<th>Title</th>
<th>Labels</th>
<th>Created</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>#17</td>
<td>Add flag to change colorscheme</td>
<td>good first issue, help wanted</td>
<td>Feb 2, 2019</td>
<td>mre</td>
</tr>
<tr>
<td>#15</td>
<td>Write an IntelliJ plugin for cargo-inspect</td>
<td>good first issue, help wanted</td>
<td>Feb 2, 2019</td>
<td>mre</td>
</tr>
<tr>
<td>#13</td>
<td>Use a random spinner while compiling</td>
<td></td>
<td>Dec 6, 2018</td>
<td>mre</td>
</tr>
<tr>
<td>#11</td>
<td>Add support for diffing two programs</td>
<td></td>
<td>Dec 1, 2018</td>
<td>mre</td>
</tr>
<tr>
<td>#10</td>
<td>Support line ranges</td>
<td>good first issue, help wanted</td>
<td>Nov 30, 2018</td>
<td>mre</td>
</tr>
<tr>
<td>#8</td>
<td>Add support for not expanding macros</td>
<td>good first issue, help wanted</td>
<td>Nov 19, 2018</td>
<td>mre</td>
</tr>
<tr>
<td>#7</td>
<td>Add links to documentation</td>
<td>good first issue, help wanted</td>
<td>Nov 19, 2018</td>
<td>mre</td>
</tr>
</tbody>
</table>
Credits

• Stage background from freepik.com designed by starline
• Lucy with a Rocket engine
• Rustlang MIR documentation
• Rust compiler guide