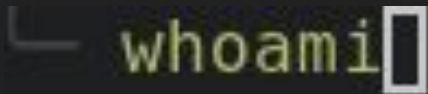


# You too can contribute to rustc!

Shrey Somaiya

(they / them)



- UNSW (2020 - Pres)
  - Intro Programming Course (COMP1511)
  - Design/Propose/Write Rust Course (COMP6991)
  - Rust/WASM teaching tools (mipsy web - yew)
- Atlassian (2021 - Pres)
  - Search UI
  - Editor UI
  - Jira Frontend Platform (developer productivity)

1 - find issue



jruderman commented on Oct 22, 2022 • edited by rustbot ▾

Contributor



Given the following code ([playground](#)):

```
fn<T> id(x: T) -> T { x }
```

The current output is:

```
error: expected identifier, found `<`
--> src/lib.rs:1:3
  |
1 | fn<T> id(x: T) -> T { x }
  |   ^ expected identifier
```

Ideally the output should look also include:

```
help: place the generic parameter list after the function name:
  |
1 | fn id<T>(x: T) -> T { x }
  |     ~~~
```

This is an easy mistake to make because `impl` items *do* require the generic parameters to go immediately after the keyword.



jruderman added [A-diagnostics](#) [T-compiler](#) labels on Oct 22, 2022

2. Find the relevant area

# Goal - find what does this!

```
error: expected identifier, found `<`  
--> spanishpear/enum_1.rs:1:7  
1 | struct<T> Point {  
  |           ^ expected identifier
```


```
> wc -l src/**/*.rs | tail -1  
902191 total
```

What if i just....





```
rustc +stage1 spanishpear/enum_1.rs -Ztreat-err-as-bug 2
```

thread 'rustc' panicked at  reporting due to '-Z treat-err-as-bug=1', compiler/rustc\_errors/src/lib.rs:1705:30

stack backtrace:

```
0: rust_begin_unwind
1: core::panicking::panic_fmt
2: <rustc_errors::Handler>::panic_if_treat_err_as_bug
3: <rustc_errors::Handler>::emit_diagnostic::{closure#2}
4: <rustc_errors::Handler>::emit_diagnostic
5: <rustc_errors::Handler>::emit_diagnostic
6: rustc_parse::new_parser_from_file
7: rustc_parse::parse_crate_from_file
8: <rustc_session::session::Session>::time::<core::result::Result<rustc_ast::ast::Crate, rustc_errors::diagnostic_builder::DiagnosticBuilder<rustc_session::ErrorGuaranteed>>, rustc_interface::passes::parse::{closure#0}>
9: rustc_interface::passes::parse
10: <rustc_interface::queries::Query<rustc_ast::ast::Crate>>::compute::<<rustc_interface::queries::Queries>::parse::{closure#0}>
11: <rustc_interface::interface::Compiler>::enter::<rustc_driver_impl::run_compiler::{closure#1}::{closure#2}, core::result::Result<core::option::Option<rustc_interface::queries::Linker>, rustc_span::ErrorGuaranteed>>
12: rustc_span::with_source_map::<core::result::Result<(), rustc_span::ErrorGuaranteed>, rustc_interface::interface::run_compiler<core::result::Result<(), rustc_span::ErrorGuaranteed>, rustc_driver_impl::run_compiler::{closure#1}::{closure#0}>::{closure#0}>
13: <scoped_tls::ScopedKey<rustc_span::SessionGlobals>>::set::<rustc_interface::interface::run_compiler<core::result::Result<(), rustc_span::ErrorGuaranteed>, rustc_driver_impl::run_compiler::{closure#1}::{closure#0}>, core::result::Result<(), rustc_span::ErrorGuaranteed>>
note: Some details are omitted. Run with 'RUST_BACKTRACE=full' for a verbose backtrace.
```



```
/// Parses a source module as a crate. This is the main entry point for the parser.
pub fn parse_crate_mod(&mut self) -> Result<'a, ast::Crate> {
    let (attrs, items, spans) = self.parse_mod(&token::Eof)?; ThinVec<Attribute>, ThinVec<P<Item>>, ModS
    Ok(ast::Crate { attrs, items, spans, id: DUMMY_NODE_ID, is_placeholder: false })
}
```

```
// Public for rustfmt usage.  
pub fn parse_ident(&mut self) -> PResult<'a, Ident> {  
    self.parse_ident_common(true)  
}
```

```
fn parse_ident_common(&mut self, recover: bool) -> PResult<'a, Ident> {  
    let (ident, is_raw) = self.ident_or_err()?; Ident, bool  
    if !is_raw && ident.is_reserved() {  
        let mut err = self.expected_ident_found(); DiagnosticBuilder<ErrorGuaranteed>  
        if recover {  
            err.emit();  
        } else {  
            return Err(err);  
        }  
    }  
    self.bump();  
    Ok(ident)  
}
```

Apply advanced debugging



```
1 pub(super) fn expected_ident_found(&mut self) -> DiagnosticBuilder<'a, ErrorGuaranteed> {  
+ 288     debug!("HERE")  
    }
```

```
DEBUG rustc_parse::parser::diagnostics HERE
```

3. Make changes!



```
DEBUG rustc_parse::parser::diagnostics: hint
DEBUG rustc_parse::parser::diagnostics: expected_ident_found: Token { kind: Lt, span: spanishpear/enum_1.rs:1:7: 1:8 (#0) }
DEBUG rustc_parse::parser::ast::lt: (increment) count=1
```

1. If the current token is a <
2. Check this is a valid place to throw a hint (i.e. function, struct, enum, trait, etc)
3. If the generics parse as valid, show a suggestion to move them!
  - a. If there are multiple generics do nothing
  - b. If the generics dont parse, do original ident error

# Tada!!!!

## (just find the PR)

```
error: expected identifier, found `<`  
--> spanishpear/enum_1.rs:1:7
```

```
1 | struct<T> Point {  
    ^ expected identifier
```

```
help: place the generic parameter name after the struct name
```

```
1 - struct<T> Point {  
1 + struct Point<T> {
```

```
error: aborting due to previous error
```

# Not Pictured

1. Learning how to do logs
2. Figuring out how to emit a suggestion (what methods to use)
3. Conversions between tokens identifiers etc (what methods to use)
4. How to lookahead in the parser
5. Writing tests!
6. X fmt
7. X test
8. X tidy
9. All the random uncertainty and waiting for help and reviews