The Rust Book

NONE

November 3, 2022

Contents

1 Programming a Guessing Game

```
use std::io;

fn main() {
    println!("Guess the number!");

    println!("Please input your guess.");

    let mut guess = String::new();

    io::stdin()
        .read_line(&mut guess)
        .expect("Failed to read line");

    println!("You guessed: {guess}");
}
```

Result's variants are `Ok` and `Err`. The `Ok` variant indicates the operation was successful, and inside `Ok` is the successfully generated value. The `Err` variant means the operation failed, and Err contains information about how or why the operation failed.

An instance of `Result` has an expect method that you can call. If this instance of `Result` is an `Err` value, expect will cause the program to crash and display the message that you passed as an argument to expect.

If the `read_line` method returns an `Err`, it would likely be the result of an error coming from the underlying operating system. If this instance of `Result` is an `Ok` value, expect will take the return value that `Ok` is holding

and return just that value to you so you can use it. In this case, that value is the number of bytes in the user's input.

If you don't call `expect`, the program will compile, but you'll get a warning: