stdlib | Macros

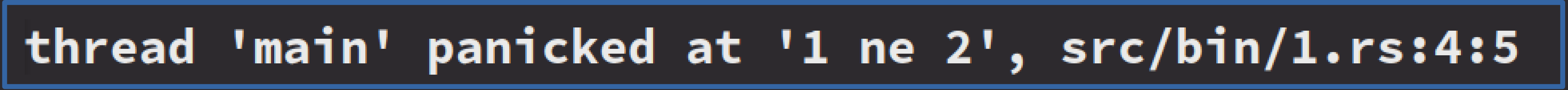
assert

```rust
let a = 1;
let b = 2;
assert!(a == b, "{} ne {}", a, b);
assert_eq!(a, b, "values should be equal");
assert_ne!(a, b, "values should not be equal");

debug_assert!(a == b, "{} ne {}", a, b);
debug_assert_eq!(a, b, "values should be equal");
debug_assert_ne!(a, b, "values should not be equal");
```

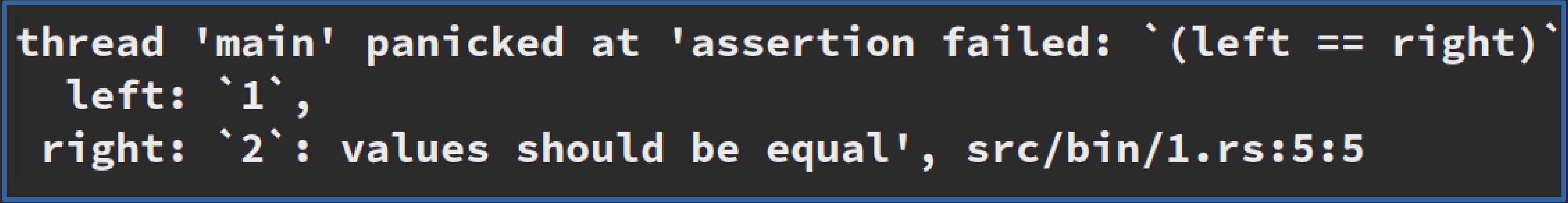
assert - output

assert!



```
thread 'main' panicked at '1 ne 2', src/bin/1.rs:4:5
```

assert_eq!



```
thread 'main' panicked at 'assertion failed: `(left == right)`
  left: `1`,
 right: `2`: values should be equal', src/bin/1.rs:5:5
```

dbg

```rust
#[derive(Debug)]
enum RoomType {
    Bedroom,
    Kitchen,
}
#[derive(Debug)]
struct Room {
    dimensions: (usize, usize),
    kind: RoomType,
}
```

```rust
let kitchen = Room {
    dimensions: (20, 20),
    kind: RoomType::Kitchen,
};
dbg!(&kitchen);
```

dbg - output

```
[src/bin/2.rs:16] &kitchen = Room {
    dimensions: (
        20,
        20,
    ),
    kind: Kitchen,
}
```

format

```rust
let h = "Hello";
let w = "World";
let greet: String = format!("{}, {}!", h, w);
println!("{}", greet);
```

include_str

- **msg.txt:** This is a message

```rust
let msg = include_str!("msg.txt");
println!("{}", msg);
```

- Data file path is relative to the source file

include_bytes

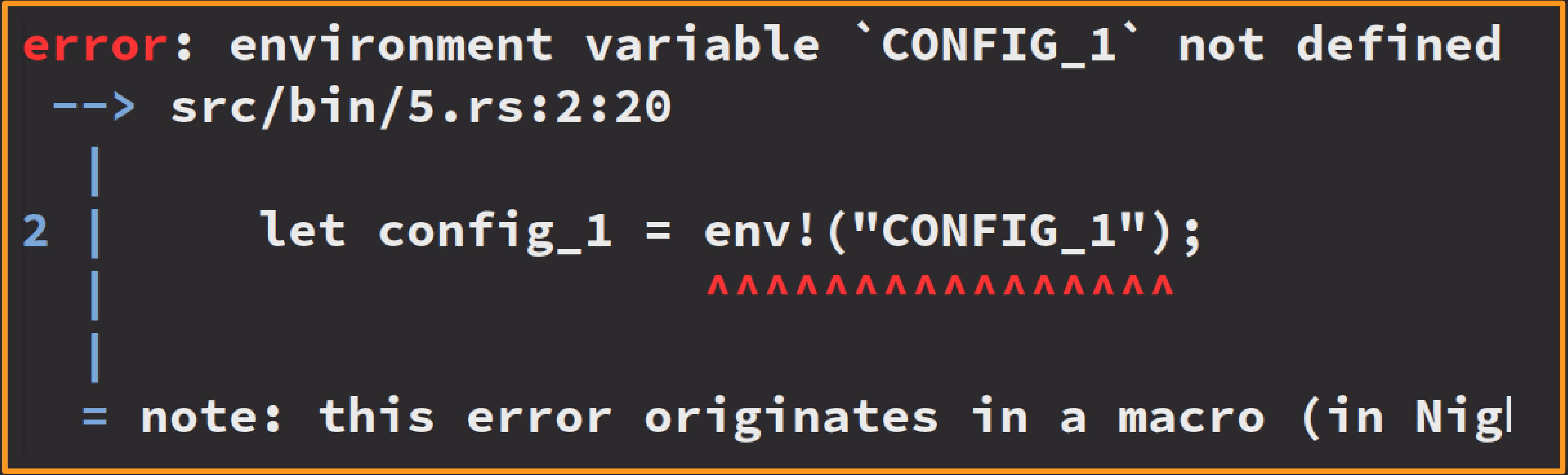
- Data is saved as an array of bytes (u8)

```
let bytes = include_bytes!("image.png");
```

env

- Include string data at compile time, based on environment variable

```rust
let config_1 = env!("CONFIG_1");
```



```
error: environment variable `CONFIG_1` not defined
 --> src/bin/5.rs:2:20
  |
2 |     let config_1 = env!("CONFIG_1");
  |                    ^^^^^^^^^^^^^^^^
  |
  = note: this error originates in a macro (in Nig
```

todo / unimplemented

- ***todo!***
  - Incomplete code sections, with intent to implement
- ***unimplemented!***
  - Incomplete code sections, with <u>no</u> intent to implement
- Program will panic when line is executed

```
todo!("taking a vacation");
unimplemented!("nobody wants this");
```

unreachable

- Indicates that some code should never be executed
  - Useful as both a debugging tool and to ease working with ***match*** arms
- Will ***panic*** at runtime if the macro is executed

unreachable - example

```rust
let number = 12;
let max_5 = {
    if number > 5 {
        5
    } else {
        number
    }
};
match max_5 {
    n @ 0..=5 => println!("n = {}", n),
    _ => unreachable!("n > 5. this is a bug"),
}
```

Recap

- ***assert*** is used to confirm if something is true
- ***dbg*** can be used to inspect values while coding
- ***format*** provides string interpolation
- ***include_str*** & ***include_bytes*** copy data from a file into the compiled binary
- ***env*** copies an environment variable into the binary
- ***todo*** indicates unfinished code
- ***unimplemented*** indicates code that will not be finished
- ***unreachable*** indicates code that should never execute