What is rustc?

Welcome to "The rustc book"! `rustc` is the compiler for the Rust programming language, provided by the project itself. Compilers take your source code and produce binary code, either as a library or executable.

Most Rust programmers don't invoke `rustc` directly, but instead do it through Cargo. It's all in service of `rustc` though! If you want to see how Cargo calls `rustc`, you can

```
$ cargo build --verbose
```

And it will print out each `rustc` invocation. This book can help you understand what each of these options does. Additionally, while most Rustaceans use Cargo, not all do: sometimes they integrate `rustc` into other build systems. This book should provide a guide to all of the options you'd need to do so.

Basic usage

Let's say you've got a little hello world program in a file `hello.rs`:

```
fn main() {
    println!("Hello, world!");
}
```

To turn this source code into an executable, you can use `rustc`:

```
$ rustc hello.rs
$ ./hello # on a *NIX
$ .\hello.exe # on Windows
```

Note that we only ever pass `rustc` the *crate root*, not every file we wish to compile. For example, if we had a `main.rs` that looked like this:

```
mod foo;

fn main() {
    foo::hello();
}
```

And a `foo.rs` that had this:

```
pub fn hello() {
    println!("Hello, world!");
}
```

To compile this, we'd run this command:

```
$ rustc main.rs
```

No need to tell `rustc` about `foo.rs`; the `mod` statements give it everything that it needs. This is different than how you would use a C compiler, where you invoke the compiler on each file, and then link everything together. In other words, the *crate* is a translation unit, not a particular module.