**Separating Modules into Different Files**

When modules get large, you might want to move their definitions to a separate file to make the code easier to navigate.

src/lib.rs

mod front\_of\_house {

pub mod hosting {

pub fn add\_to\_waitlist() {}

}

}

pub use crate::front\_of\_house::hosting;

pub fn eat\_at\_restaurant() {

hosting::add\_to\_waitlist();

}

We will spilt above code into multiple files.

First, we’ll extract the front\_of\_house module to its own file. Remove the code inside the curly brackets for the front\_of\_house module, leaving only the mod front\_of\_house; declaration, so that src/lib.rs contains the code shown in Listing 7-21. Note that this won’t compile until we create the src/front\_of\_house.rs file

src/lib.rs

mod front\_of\_house;

pub use crate::front\_of\_house::hosting;

pub fn eat\_at\_restaurant() {

hosting::add\_to\_waitlist();

}

src/front\_of\_house.rs

pub mod hosting {

pub fn add\_to\_waitlist() {}

}

Next, we’ll extract the hosting module to its own file. The process is a bit different because hosting is a child module of front\_of\_house, not of the root module. We’ll place the file for hosting in a new directory that will be named for its ancestors in the module tree, in this case src/front\_of\_house/.

To start moving hosting, we change src/front\_of\_house.rs to contain only the declaration of the hosting module:

src/front\_of\_house.rs

pub mod hosting;

src/front\_of\_house/hosting.rs

pub fn add\_to\_waitlist() {}

Note that you only need to load a file using a mod declaration once in your module tree. Once the compiler knows the file is part of the project (and knows where in the module tree the code resides because of where you’ve put the mod statement), other files in your project should refer to the loaded file’s code using a path to where it was declared, as covered in the “Paths for Referring to an Item in the Module Tree” section.

**In other words, mod is not an “include” operation that you may have seen in other programming languages.**

# Alternate File Paths

Rust also supports an older style of file path. For a module named front\_of\_house declared in the crate root, the compiler will look for the module’s code in:

src/front\_of\_house.rs (what we covered)

src/front\_of\_house/mod.rs (older style, still supported path)

For a module named hosting that is a submodule of front\_of\_house, the compiler will look for the module’s code in:

src/front\_of\_house/hosting.rs (what we covered)

src/front\_of\_house/hosting/mod.rs (older style, still supported path)

If you use both styles for the same module, you’ll get a compiler error. Using a mix of both styles for different modules in the same project is allowed, but might be confusing for people navigating your project.

# References

<https://doc.rust-lang.org/book/ch07-05-separating-modules-into-different-files.html>

# END OF FILE