##### 一、Cargo的使用

//1.较大的程序编译工具推荐使用 Cargo

// （1）Cargo是rust的构建系统和包管理工具 ---其作用可以用于构建代码、下载依赖的库、构建这些库

// （2）安装ruts的时候会自动安装Cargo ----还是可以使用 cargo --version 命令进行检测是否安装

//2.使用这个工具进行项目的创建

// （1）创建项目的命令格式： cargo new xxxxxx ---xxx代表的是项目名

// （2）会自动生成一个文件夹，文件夹包含着一个子文件夹 src ，以及如下三个默认生成的文件。

//src文件中包含着 main.rs 也就是源代码文件

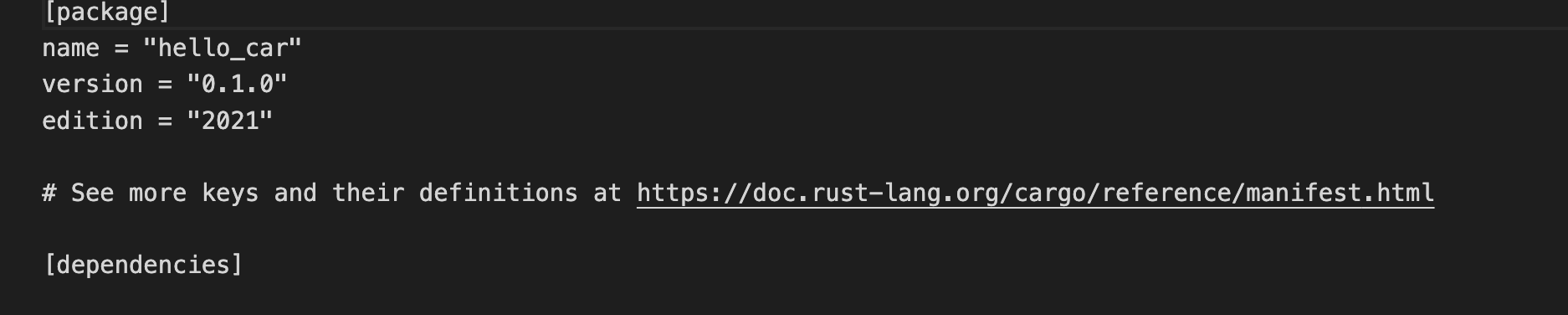
//三个默认生成的文件是 Cargo.lock Cargo.toml gitignore

//gitignore 说明了创建的时候初始化了一个仓库

//其余的两个是因为安装了这个插件默认生成的两个文件

//如果不想生成这个仓库就使用命令 cargo new --vcs xxxxx 就不会生成这个仓库文件了

3.Cargo.toml文件



（1）.toml 是一种文本文件的格式，通常作为配置文件使用

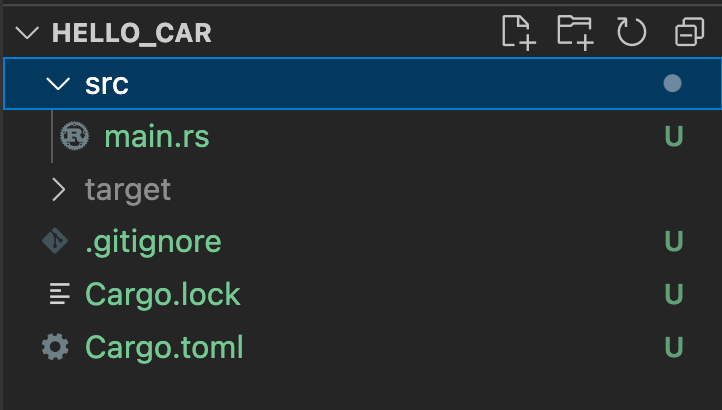
（2）[package] 是一个区域标题，下方的内容是用来配置包的 包：package

//name代表项目名 version代表项目版本 authors代表项目作者 edition:代表使用的rust版本

//[dependencies] ：另一个区域的开始。会列出项目的依赖项

（3）在rust里面，代码的包称作create

4. src/main.rs



（1）cargo生成的main.rs 在src目录的下面

（2）而Cargo.toml 应该在项目顶层下面，源代码都应该在src目录下面 ---项目顶层的意思就是打开helo\_car项目就能看见这个文件

（3）项目顶层可以放置：README、许可信息、配置文件、和程序源码无关的文件

（4）如果创建项目的时候没有使用cargo，也可以把项目转化为使用cargo

//将源代码文件移动到src下面 创建Cargo/toml并填写相应的配置

##### 二、使用Cargo build构建项目

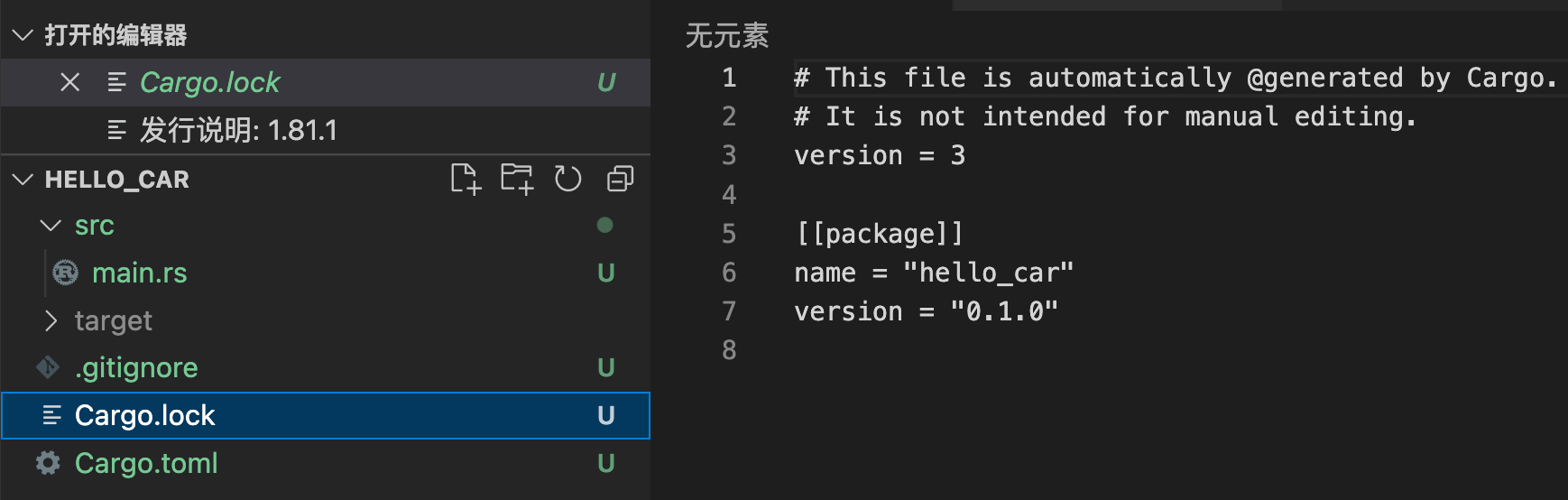
//对某个项目第一次使用cargo build构建项目的时候，会生成一个文件夹target，这个文件夹里面包含着debug文件夹

1.使用 cargo build 这个命令会创建一个可执行文件： ----这个命令后面跟随的就是项目名字，创建完项目之后就会生成一个可执行文件

（1）文件路径：target/debug/hello\_cargo 或者target\debug\hello\_cargo.exe (windows下生成的)

（2）运行可执行文件的方法和Linux一样

（3）第一次运行 cargo build 会在顶层目录生成 cargo.lock文件



//该文件负责追踪项目依赖的精确版本 并且不需要手动修改该文件

2.如果想要生成可执行文件，并且是让用户使用的，那么就可以使用 cargo build --release

（1）编译的时候会进行优化，代码会运行的更快，但是编译的时间会更长

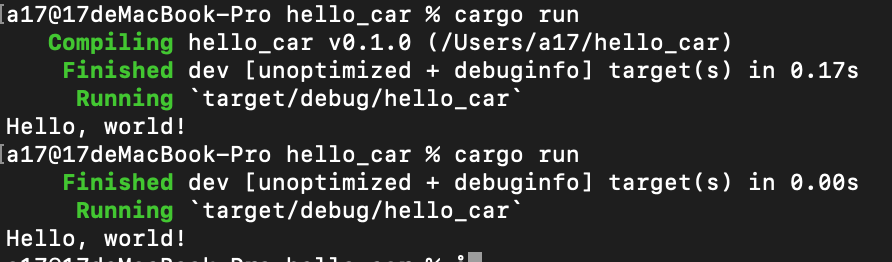
（2）会在target/release 而不是 target/debug 这个目录下生成可执行文件

##### 三、构建和运行cargo项目

、1.使用cargo run 命令 ：可以构建并且运行cargo项目 ----可以说cargo run 可以运行项目

（1）所以这个命令是有两步操作的，编译代码+执行结果

（2）如果之前编译成功过，并且源码没有改变，那么就会直接运行二进制文件



##### 四、cargo check

1.cargo check ：检查代码，确保能通过编译，但是不产生任何可执行文件

2. cargo check 比 cargo build 快的多

//编写代码的时候可以连续反复的使用cargo check检查代码，提高效率

3.如果准备发布构建，就是让用户使用而想要生成可执行文件，使用 cargo build --release

4.两个配置，一个是开发用的，一个是正式发布用的

//补充一点：外部的包就是crate crate：箱子