

## Compilación y Ejecución en Rust

La ejecución de código en Rust puede realizarse mediante dos vías fundamentales:

- Compilador directo, rustc, para tareas sencillas.
- Cargo la herramienta estándar de gestión de proyectos, indispensable para el desarrollo moderno.

### Usando rustc

Para comprender la esencia del proceso de compilación, es crucial familiarizarse con rustc, el compilador de Rust. Este método es ideal para archivos individuales o para entender cómo el código fuente se traduce en un ejecutable binario.

#### Fases del Proceso rustc

##### 1. Paso 1: Creación del Módulo Fuente

Todo comienza con el código fuente, que tradicionalmente lleva la extensión `.rs`.

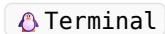
```
fn main() {  
    println!("Compilador directo rustc."); //Archivo: main.rs  
}
```



##### 2. Paso 2: Compilación

Desde la terminal, se invoca a rustc, apuntando al archivo de entrada. El compilador lee el código y genera un archivo binario ejecutable en el mismo directorio.

```
rustc main.rs
```



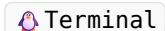
En este proceso, rustc maneja internamente la verificación de tipos, el borrow checker y la generación del código máquina optimizado, utilizando LLVM.

##### 3. Ejecución

Esto genera un ejecutable.

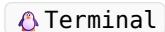
- Windows:

```
.\main.exe
```



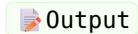
- Linux/macOS:

```
./main
```



Resultado:

```
Compilador directo rustc.
```



## Tu Primer Proyecto con Cargo

Crear un nuevo proyecto

```
# Crear un proyecto binario (aplicación)
```



```
cargo new a hola_mundo b
```

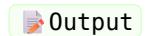
- a: Crear un nuevo proyecto Rust
- b: Nombre del proyecto

```
# Entrar al directorio  
cd hola_mundo
```



Estructura creada:

```
hello_world/ Raíz del proyecto  
└── Cargo.lock Registra las versiones específicas  
└── Cargo.toml Define las dependencias  
└── src/  
    └── main.rs Código fuente principal  
└── target/ Destino de la Compilación  
    └── debug/  
        └── build/  
        └── deps/  
            └── hello_world El ejecutable de tu aplicación
```



Anatomía del Proyecto: Cargo.toml

Contenido inicial de Cargo.toml

```
[package] toml  
name = "hola_mundo" Nombre del proyecto  
version = "0.1.0" Versión siguiendo  
edition = "2021" Edición estable de Rust 2021
```

```
[dependencies] crates  
# Ejemplos  
# rand = "0.8.5" Permite generar números aleatorios  
# serde = "1.0.130" Permite serializar y deserializar datos
```



} (1)  
Metadatos

} (2)  
Liberías externas

Punto de entrada: main.rs

```
fn main() {  
    println!("Hola, Rust!");  
}
```

