



链滴

Rust 中的注释

作者: [lingyundu](#)

原文链接: <https://ld246.com/article/1612057207430>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

Rust 中的注释分为两种：

- 普通注释 -- 仅做注释用，在编译时编译器会忽略它们。
- 文档注释 -- 可以通过命令生成 HTML 帮助文档。

普通注释

Rust 的普通注释与 C++ 的风格一样，分为：

- 单行注释 -- 以 `//` 开头，`//` 后的内容都会被注释掉。
- 块注释 -- 可以注释多行，并且可以嵌套，使用 `/* ... */` 将注释内容与代码分隔。

示例：

```
fn main() {
    // 这是行注释

    /* 这是块注释，
    可以注释多行。*/

    /*
    * 这也是块注释
    * 该行前面的星号不是必须的
    * 但这样比较好看
    */

    /* 块注释可以嵌套 */ 嵌套有什么用? */ 我很疑惑 */

    // 块注释可以注释掉一行中间的部分代码，行注释则没有这个能力
    let x = 5 + /* 90 + */ 5;
    println!("Is `x` 10 or 100? x = {}", x);
}
```

文档注释

文档注释也分为单行注释和块注释，但又有内外之分：

- 内部文档注释 (Inner doc comment)
 - 单行注释 (以 `///` 开头)
 - 块注释 (用 `/** ... */` 分隔)
- 外部文档注释 (Outer doc comment)
 - 单行注释 (以 `/*!` 开头)
 - 块注释 (用 `/*! ... */` 分隔)

二者的区别：

- 内部文档注释是对它 **之后**的项做注释，与使用 `#[doc="..."]` 是等价的。
- 外部文档注释是对它 **所在**的项做注释，与使用 `#![doc="..."]` 是等价的。

另外，在文档注释中可以使用 Markdown 语法。

示例：

使用 `cargo new comment --lib` 创建一个项目，将下面的代码 `src/lib.rs` 是文件中的内容。

```
//! # Comment
//! 外部文档注释--描述包含它的项，一般用来编写 Crate 的文档注释。

/// - 内部文档注释 -- `outer_module` 的单行注释1
/** - 内部文档注释 -- `outer_module` 的块注释1 */
pub mod outer_module {

    //! - 外部文档注释 -- `outer_module` 的单行注释2
    /*! - 外部文档注释 -- `outer_module` 的块注释2 */


    /// - 内部文档注释 -- `inner_module` 的单行注释
    /** - 内部文档注释 -- `inner_module` 的块注释 */
    pub mod inner_module {}

    /// 这是代表人类一个结构体
    pub struct Person {
        /// 一个人必须有名字
        name: String,
    }

    impl Person {
        /// 返回具有指定名字的一个人
        ///
        /// # 参数
        ///
        /// * `name` - 字符串切片，代表人的名字
        ///
        /// # 示例
        ///
        /// ```
        /// // 在文档注释中，可以编写代码块
        /// // 如果向 Rustdoc 传递 --test 参数，它还会帮你测试注释文档中的代码!
        /// use comment::Person;
        /// let person = Person::new("name");
        /// ```
        pub fn new(name: &str) -> Person {
            Person {
                name: name.to_string(),
            }
        }
    }
}
```

使用 `cargo doc` 命令可以生成 Rust 项目的 HTML 帮助文档，上面的示例生成的文档如下图所示：

comment Crate 的文档：


 All crates ▾ Click or press 'S' to search, '?' for more options... ? ⓘ

Crate comment [-][src]

[-] **Comment**

外部文档注释--描述包含它的项，一般用来编写 Crate 的文档注释。

\$Modules

`outer_module`

- 内部文档注释 -- `outer_module` 的单行注释1
- 内部文档注释 -- `outer_module` 的块注释1
- 外部文档注释 -- `outer_module` 的单行注释2
- 外部文档注释 -- `outer_module` 的块注释2

outer_module 模块的文档:


 All crates ▾ Click or press 'S' to search, '?' for more options... ? ⓘ

Module comment::outer_module [-][src]

[-]

- 内部文档注释 -- `outer_module` 的单行注释1
- 内部文档注释 -- `outer_module` 的块注释1
- 外部文档注释 -- `outer_module` 的单行注释2
- 外部文档注释 -- `outer_module` 的块注释2

\$Modules

`inner_module`

- 内部文档注释 -- `inner_module` 的单行注释
- 内部文档注释 -- `inner_module` 的块注释

Structs

`Person` 这是代表人类一个结构体

Person 结构体的文档:


 All crates ▾ Click or press 'S' to search, '?' for more options... ? ⓘ

Struct comment::outer_module::Person [-][src]

[+] Show declaration

[-] 这是代表人类一个结构体

Implementations

[-] `impl Person` [src]

[-] `pub fn new(name: &str) -> Person` [src]

返回具有指定名字的一个人的

参数

- `name` - 字符串切片，代表人的名字

\$ 示例

```

// 在文档注释中，可以编写代码块
// 如果向 Rustdoc 传递 --test 参数，它还会帮你测试注释文档中的代码！
use comment::Person;
let person = Person::new("name");
  
```

Auto Trait Implementations

相关资料

[The Rust Reference](#)

[The rustdoc book](#)

[Rust By Example](#)