SUMMARY ON ARTICLE : “async/.await”

In Rust programming language the most predictable feature is finally put in, named as “async/.await”. Here I gave short summary on asynchronous programming in rust , here we go.

We understand async function by taking example of simple application that take some data from internet by using our new function’s ability ‘asyn’. Before going on coding we discuss about basic concept of asynchronous programming in rust .

**Async in rust:**

In rust async means run multiple program in one time or in single thread. It is useful for IO bound task which consumes time. For getting result of asynchronous programming we must .await it . In rust awaitable value known as “future” value.

**Rusty strangeness:**

Asynchoronous programming in rust is different from other programming language like java script, C# etc. Here is the main point of asynchoronus programming that we need to understand.

**An Async func execution:**

Async func does not execute immediately . We need .await or an executer to start it. Lets take an example.

**use async\_std::task;**

**// ^ we need this for task spawning**

**async fn negate\_async(n: i32) -> i32 {**

**println!("Negating {}", n);**

**task::sleep(std::time::Duration::from\_secs(5)).await;**

**println!("Finished sleeping for {}!", n);**

**n \* -1**

**}**

**async fn f() -> i32 {**

**let neg = negate\_async(1);**

**// ... nothing happens yet**

**let neg\_task = task::spawn(negate\_async(2));**

**// ^ this task /is/ started**

**task::sleep(std::time::Duration::from\_secs(1)).await;**

**// we sleep for effect.**

**neg.await + neg\_task.await**

**// ^ this starts the first task `neg`**

**// and waits for both tasks to finish**

**}**

**The explanation of the above code are as follow:**

1. In first line,We import external library “async\_std::task” because standard library does not provide executer.
2. The async function “negate\_async” take signed integer as input sleep for 5 sec and return negative value.
3. In Async function “f”: Let neg create future of negat\_async but does not execute. Let neg\_task uses task::spwan function to start executing future of negat\_async and assigned to neg \_task . Now sleep for a second which can see in output . now add neg and neg\_task . We start executing future and wait for completion of neg\_task.

The Output of above Task is:

Negating 2

# <- there's a 1 second pause here

Negating 1

Finished sleeping for 2!

Finished sleeping for 1!

As we see neg\_task is executed by task::spwan help while neg is not start executing until it is awaited.