22异步的原理

2020年3月9日 12:34

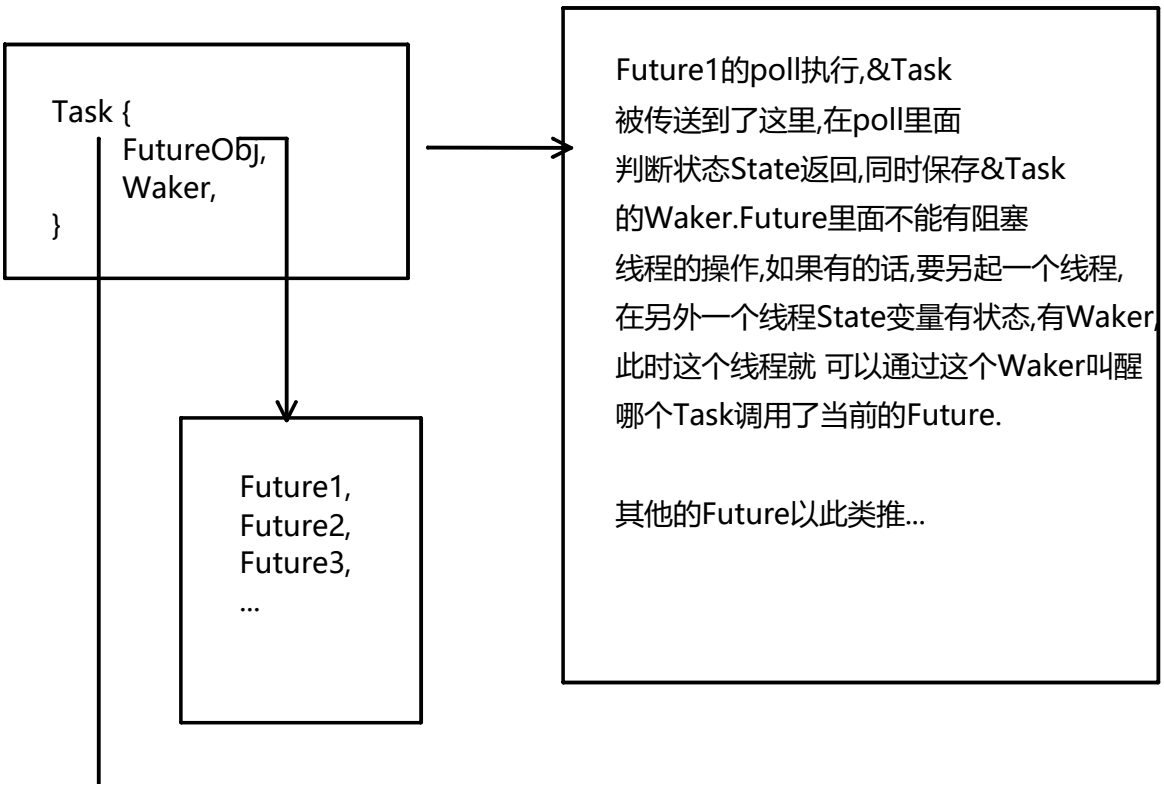
一个重要的问题:Waker到底是什么的?Task到底是什么?执行器到底是怎么执行的?

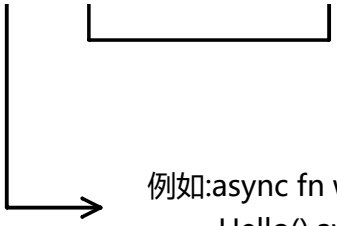
(1):Task究竟是什么?和Future的关系是什么?
一句话Task就是一个顶级的Future.

这个顶级的Future下面可能还有许多的Future,需要的Future下面还有Future.
但是Task就是那个最顶级的Future的抽象.而其他的Future都是在poll()函数
里面编译器自动添加了生成器的resume()函数驱动着Future就行执行的.

过程是这样的:
Task ->调用poll(),task对应的Future调用resume()函数,顶级的Future就会
被驱动起来执行 -> 然后按照顶级下面的Future有多少个Future进行Poll,
poll()函数也会调用resume()函数驱动次级的Future执行起来 -> 次级Future
的resume()返回GeneratorState::Yield,那么调用者返回Poll::Pending,以此来推.

(2): Task里面有一个Waker,这个是具体执行Future的代码和Task沟通的一个通道.





例如:async fn women() {

```
    Hello().await;
    Hello2().await;
    Hello3().await;
    ...
}
```

Women()这个函数返回了一个Future,Task里面的FutureObj就是这个Women()函数返回的.

Women()函数里面还有许多的Future,编译器会在poll()里面添加生成器的resume()函数调用,因为Future这个trait是#[lang="future_trait"],编译器照顾的类型.

顶层的执行器,调用Task的poll()函数,在Task里面有一个顶级的生成器,这个生成器调用resume()函数开始执行.

women()函数里面又有三个Future,也就是有三个生成器,这个三个Future在Task的resume()具体的执行代码里面又会被分别的调用poll()以及具体的resume().三个Future全部返回Ready,Task才返回Ready,否则返回Pending.