

Rust Programming Note

Keep recording ...

Preface

Contents

1	Asynchronous Programming	1
1.1	Fundamental Concepts	1
1.2	Other Concurrency Models	1
1.3	Future	1
1.3.1	Leaf futures	2
1.3.2	Non-Leaf futures	2
1.3.3	A mental model of an async runtime	2
1.4	Resource Collection	2
A	Recommendation	3

Chapter 1

Asynchronous Programming

1.1 Fundamental Concepts

Definition 1.1.1 (Asynchronous Programming). *A concurrent programming model.*

Async improve the system performance by nont blocking the threads.

It lets you run a large number of concurrent tasks on a small number of OS threads, while preserving much of the look and feel of ordinary synchronous programming, through the *async/await* syntax.

相比与传统的同步编程，异步编程可以更好地处理 IO 密集型任务和并发请求，提高系统的吞吐量核性能。

1.2 Other Concurrency Models

- OS threads
- Event-driven programming
- Coroutines
- The actor model

1.3 Future

Future is an asynchronous computation that will be completed in the future.

```
1 pub trait Future {
2     type Output;
3
4     // Require method
5     fn poll(self: Pin<&mut Self>, cx: &mut Context<'_>) -> Poll<Self::Output>;
6 }
7
8 pub enum Poll<T> {
9     Ready(T),
10    Pending,
11 }
```

Async in Rust uses a *poll-based approach* in which an asynchronous task will have three phases.

1. *The poll phase:* We often refer to the part of the runtime that polls a future as an *executor*.
2. *The wait phase:* An event source, most often referred to as a *reactor*.

3. **The wake phase:** The event happens, and the future is woken up.

1.3.1 Leaf futures

Definition 1.3.1 (Leaf futures). *Runtime create it, which represent a resource such as a socket.*

1.3.2 Non-Leaf futures

Definition 1.3.2 (Non-Leaf futures). *The kind of futures we as users of runtime write ourselves using the `async` keyword to create a task that can be run on the executor.*

1.3.3 A mental model of an async runtime

Rust doesn't come with a runtime for handling concurrency.

- **Reactor** (responsible for notifying about I/O events)
- **Executor** (scheduler)
- **Future** (a task that can stop and resume at specific points)

1.4 Resource Collection

- [Asynchronous Programming in Rust](#)
- [Rust Course](#)
- [concurrency-programming-via-rust](#)
- [Asynchronous Programming in Rust \(By Carl Fredrik Samson\)](#)
- [Hands-On-Concurrency-with-Rust](#)

Appendix A

Recommendation