Rust 编程语言入门

杨旭，微软 MVP
Rust、Go 开发者

16.2 使用消息传递来跨线程传递数据

消息传递

- 另一种很流行且能保证安全并发的技术就是：**消息传递**。
  - 线程（或 Actor）通过彼此发送消息（数据）来进行通信
- Go 语言的名言：***不要用共享内存来通信，要用通信来共享内存***。
- Rust：Channel（标准库提供）

Channel

- Channel 包含：发送端、接收端
- 调用发送端的方法，发送数据
- 接收端会检查和接收到达的数据
- 如果发送端、接收端中任意一端被丢弃了，那么 Channel 就“**关闭**”了

创建 Channel

- 使用 mpsc::channel 函数来创建 Channel
  - mpsc 表示 multiple producer, single consumer（多个生产者、一个消费者）
  - 返回一个 tuple（元组）：里面元素分别是发送端、接收端
- （例子）

发送端的 send 方法

- 参数：想要发送的数据
- 返回：Result<T, E>
  - 如果有问题（例如接收端已经被丢弃），就返回一个错误

接收端的方法

- recv 方法：阻止当前线程执行，直到 Channel 中有值被送来
  - 一旦有值收到，就返回 Result<T, E>
  - 当发送端关闭，就会收到一个错误

- try_recv 方法：不会阻塞，
  - 立即返回 Result<T, E>：
    - 有数据达到：返回 Ok，里面包含着数据
    - 否则，返回错误
  - 通常会使用循环调用来检查 try_recv 的结果

Channel 和所有权转移

- 所有权在消息传递中非常重要：能帮你编写安全、并发的代码
- （例子）

发送多个值，看到接收者在等待

- （例子）

通过克隆创建多个发送者

- （例子）

再见