多线程Web服务器

线程池的编写

1. 确定线程条数，自定义方法size
2. Mpsc::channel()方法建立线程通道，返回的一般是需要Box<dyn FnOnce()+Send+'Static>的（sender，receiver）
3. receiver的多线程处理Arc::new(Mutex::new(receiver))
4. 【这里的理解，就是要对于各种智能指针的理解和运用，比如Box适用于未知内存解决的情况，Send是多线程的所有权传递，Arc和Mutex是处理多线程的内部可变性，同时mpsc是应用于多个发送端一个接收端的情况】

对多线程函数的重构

这个是收获最大的；

```
trait FnBox {//这里将类型加Box，实现未知内存的解引用
    fn call_box(self:Box<Self>);
}

impl <F:FnOnce()> FnBox for F {//对FnOnce实现FnBox的方法重构，实现利用Box来进行解引用self
    fn call_box(self:Box<Self>) {
        (*self)()
    }
}
```

自定义FnBox的trait，实现对类型Self的Box智能指针后的解引用，也就是变相引入deref trait，因为Box实现了deref；

然后通过对F类型FnOnce实现FnBox，进而实现解引用功能；

总结

对整个服务器的发送端，接收端的流程了解；

- TCPListener::bind监听函数
- for循环处理监听后的listener.incoming()

- 对listener.incoming()的处理，将其利用read函数放入缓存，再根据表头“GET / ”等进行相应的操作，对读取内容进行跳转等一些逻辑函数；
- 多线程池的理解