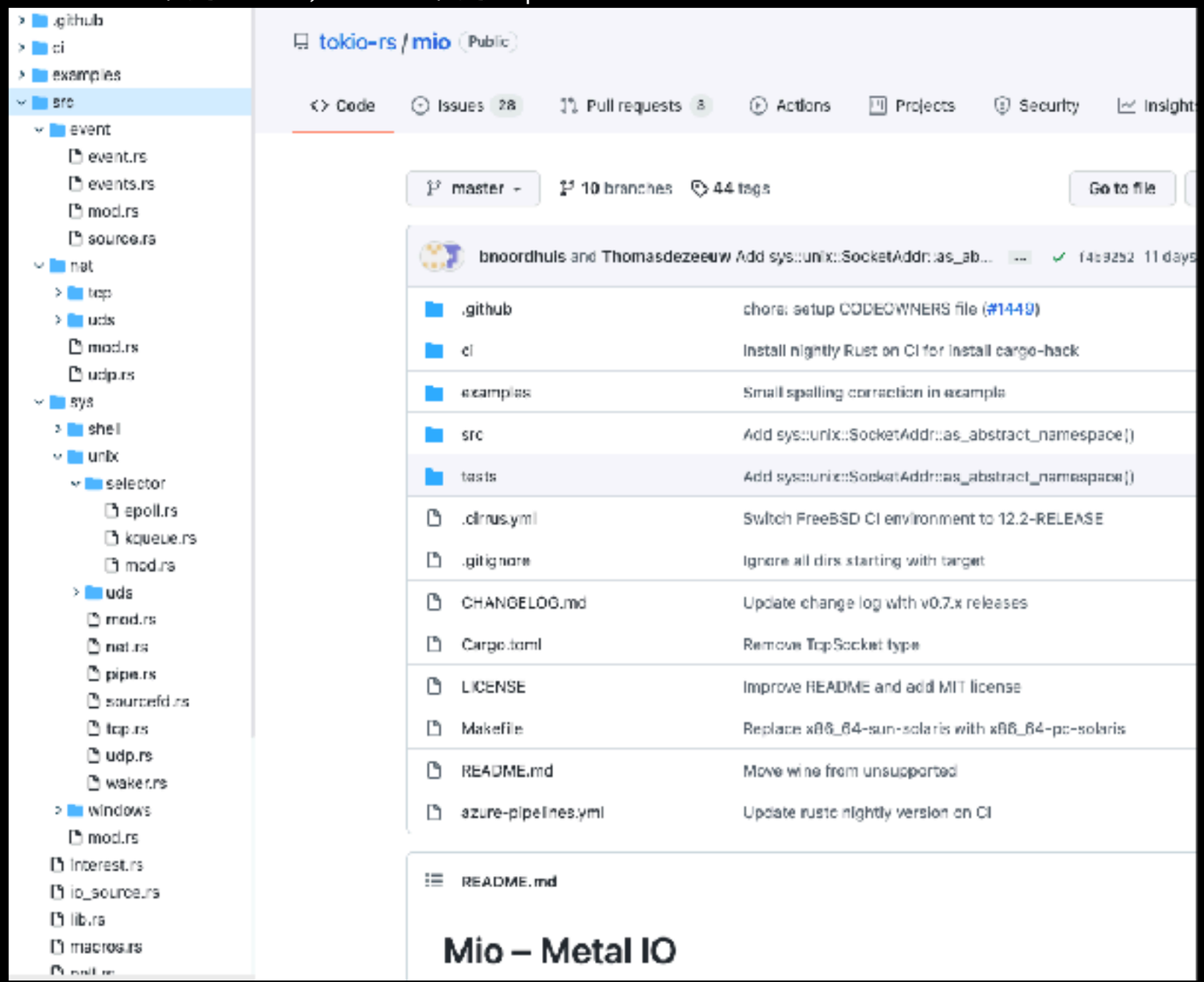
探讨Rust异步编程框架Mio

为深入理解Tokio打基础

苏林

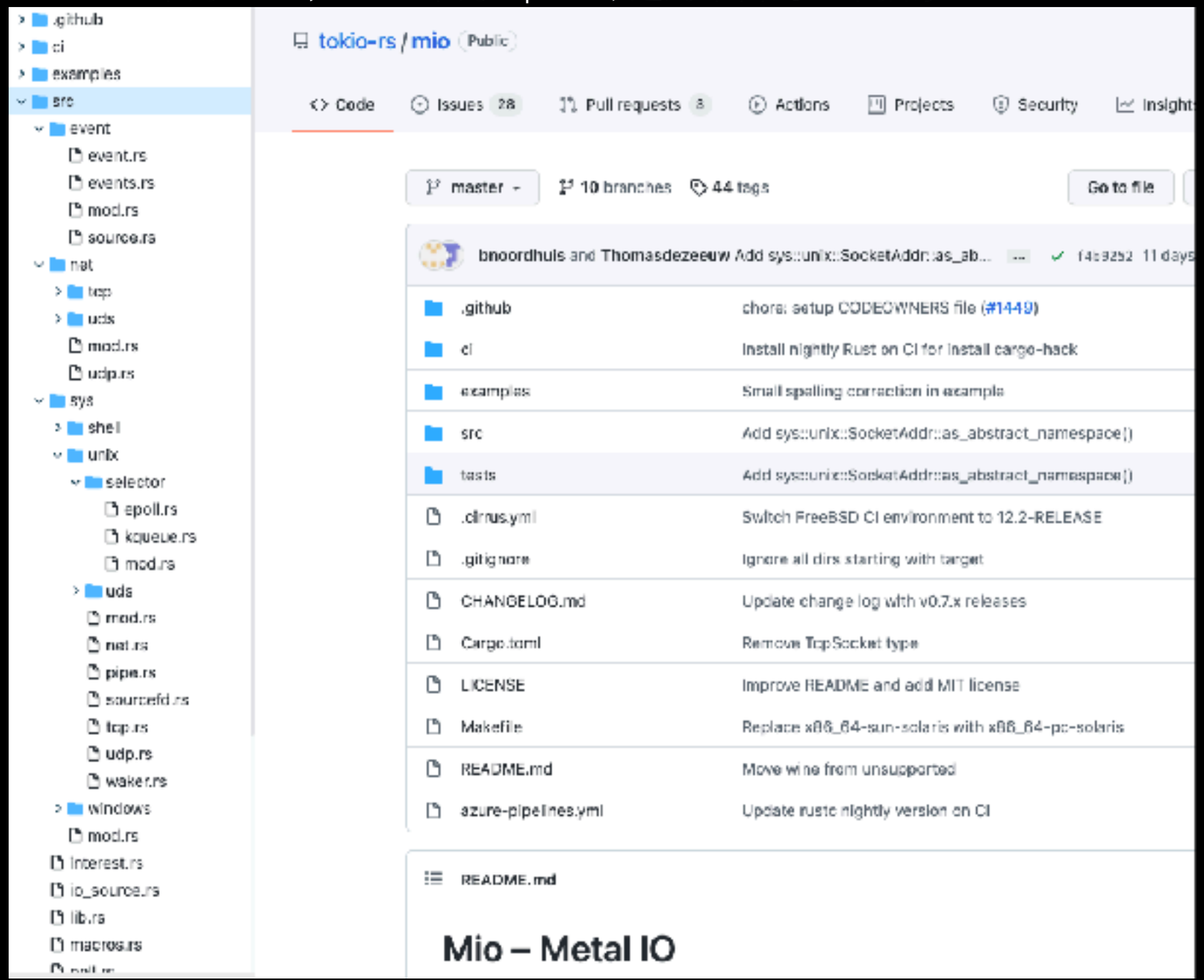
什么是Mio

mio是rust实现的一个轻量级的底层异步I/O库, 提供非阻塞方式的API, 具有很高的性能. 其实现基本上就是对不同操作系统底层相关API(epoll/kqueue/IOCP)的封装，抽象出统一的接口供上层使用。Linux下为epoll, Windows下为IOCP，OS X下为kqueue.



什么是Mio

mio是rust实现的一个轻量级的底层异步I/O库, 提供非阻塞方式的API, 具有很高的性能. 其实现基本上就是对不同操作系统底层相关API(epoll/kqueue/IOCP)的封装，抽象出统一的接口供上层使用。Linux下为epoll, Windows下为IOCP，OS X下为kqueue, 是跨平台的一个抽象.



1、跨平台的.

2、event事件模型.

3、多路复用(epll、kqueue、iocp)

对Mio所涉及的知识进行拆分

1、什么是I/0多路复用(以linux下的epoll来进行讲解).
https://github.com/zupzup/rust-epoll-example

2、什么是事件驱动(Reactor)
https://github.com/zupzup/rust-reactor-executor-example

3、如何进行跨平台抽象(examples-minimio)
https://github.com/cfsamson/examples-minimio

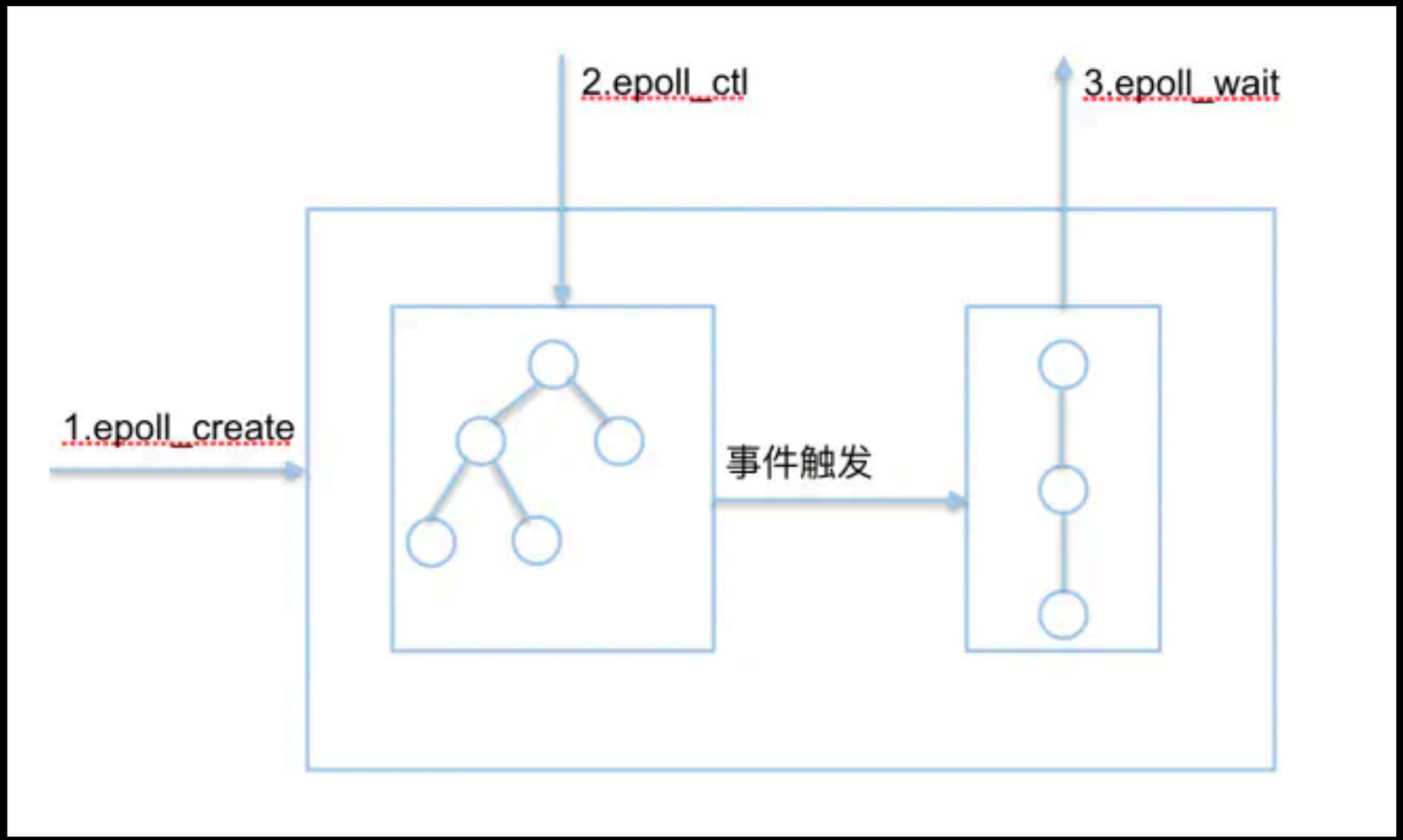
分享内容

- Rust异步编程之epoll以及rust-epoll-example库来加深理解
- Rust异步编程之事件驱动编程以及rust-reactor-executor-example库来加深理解
- 如何理解跨平台抽象以及examples-minimio库来加深理解
- 总结Mio

讨论epoll的工作原理是什么

至于为什么需要epoll这个不讲, 大家可以下来查看相关资料.

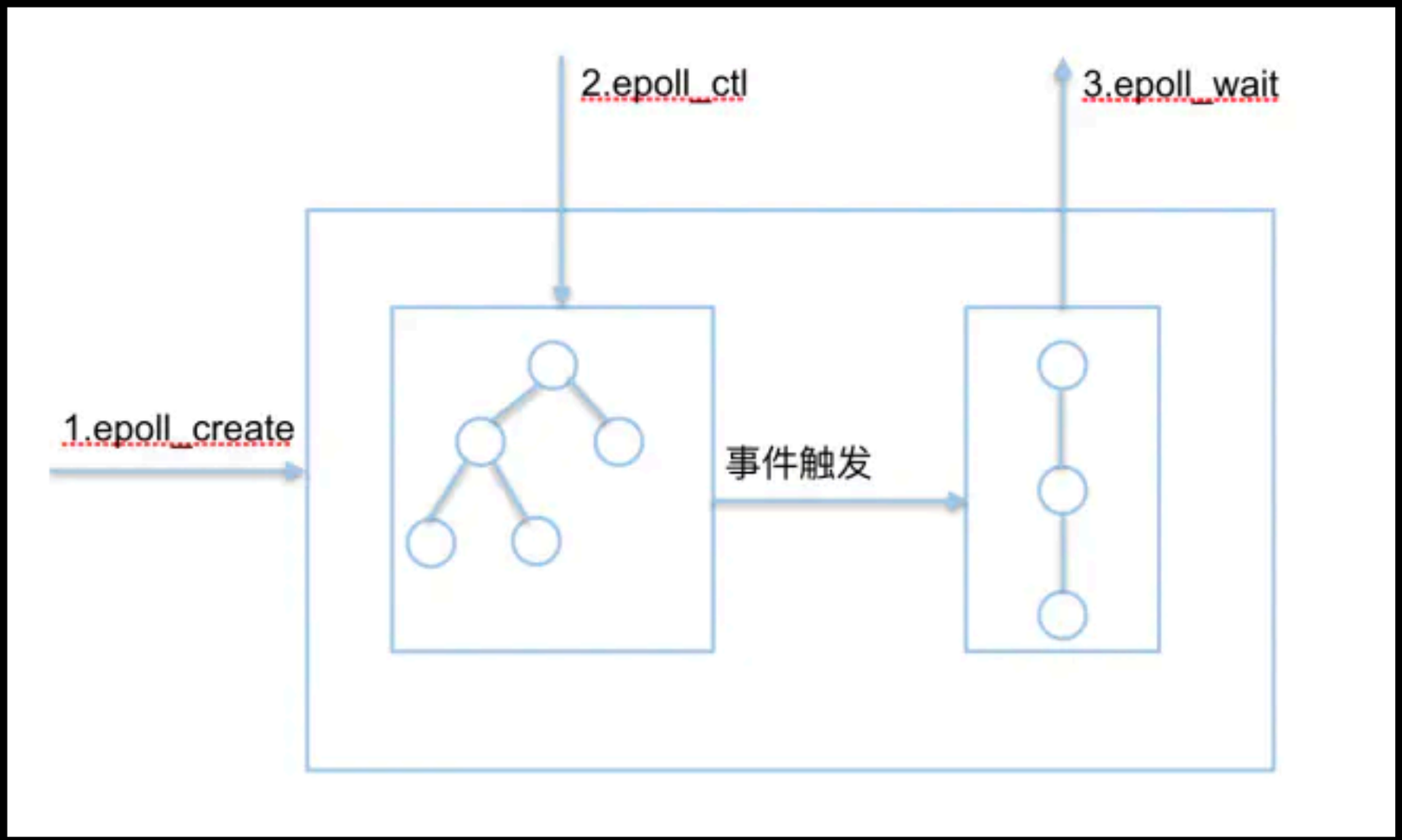
epoll的核心是3个API, 核心数据结构是: 1个红黑树和1个链表.



讨论epoll的工作原理是什么

1、libc::epoll_create1 -> unsafe extern "C" fn epoll_create1(flags: c_int) -> c_int

内核会产生一个epoll 实例数据结构并返回一个文件描述符，这个特殊的描述符就是epoll实例的句柄，后面的两个接口都以它为中心（即epfd形参）



讨论epoll的工作原理是什么

2、int epoll_ctl -> unsafe extern "C" fn epoll_ctl(epfd: c_int, op: c_int, fd: c_int, event: *mut epoll_event) -> c_int

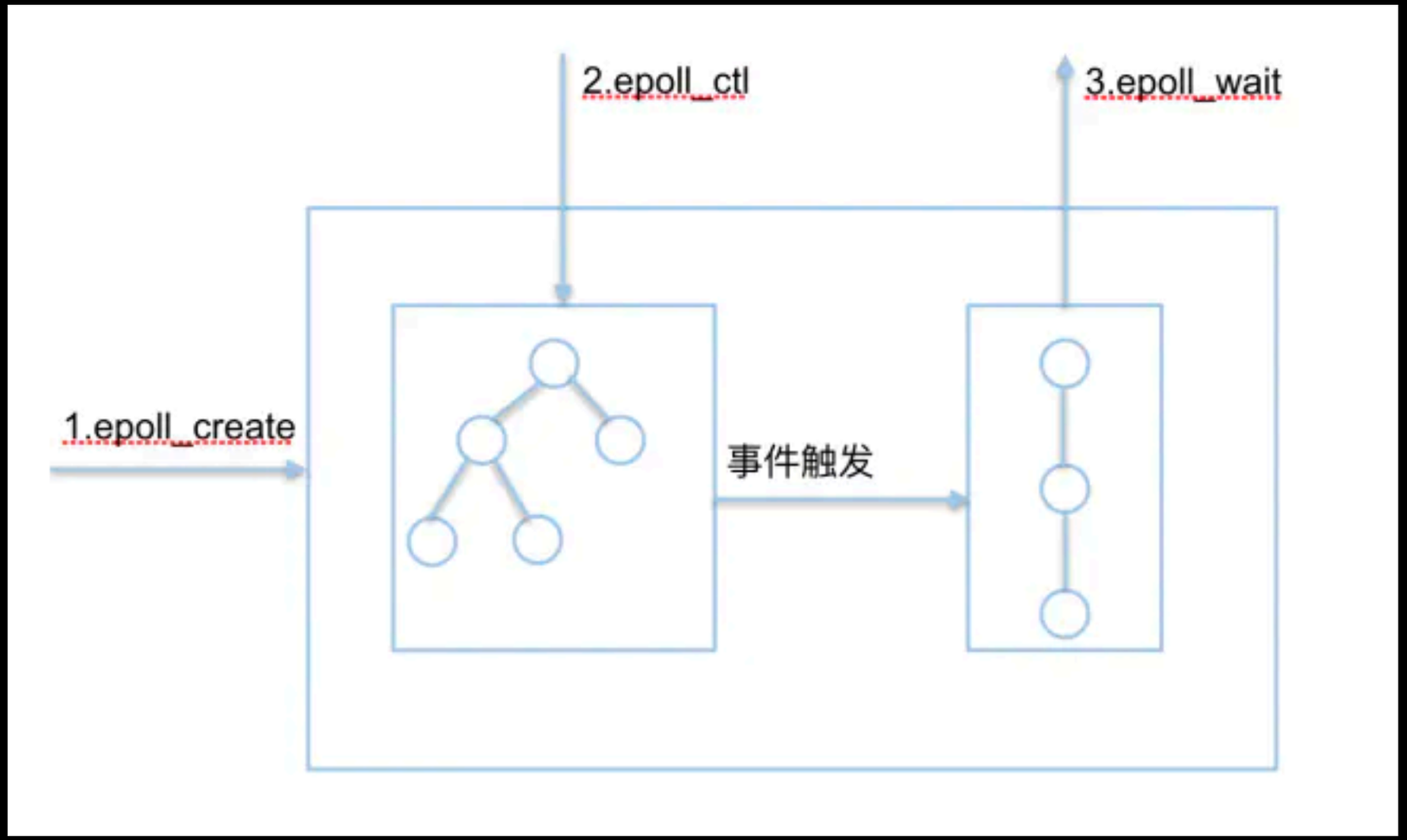
将被监听的描述符添加到红黑树或从红黑树中删除或者对监听事件进行修改

op参数说明操作类型：

EPOLL_CTL_ADD：向红黑树中添加一个需要监视的描述符

EPOLL_CTL_DEL：从红黑树中删除一个描述符

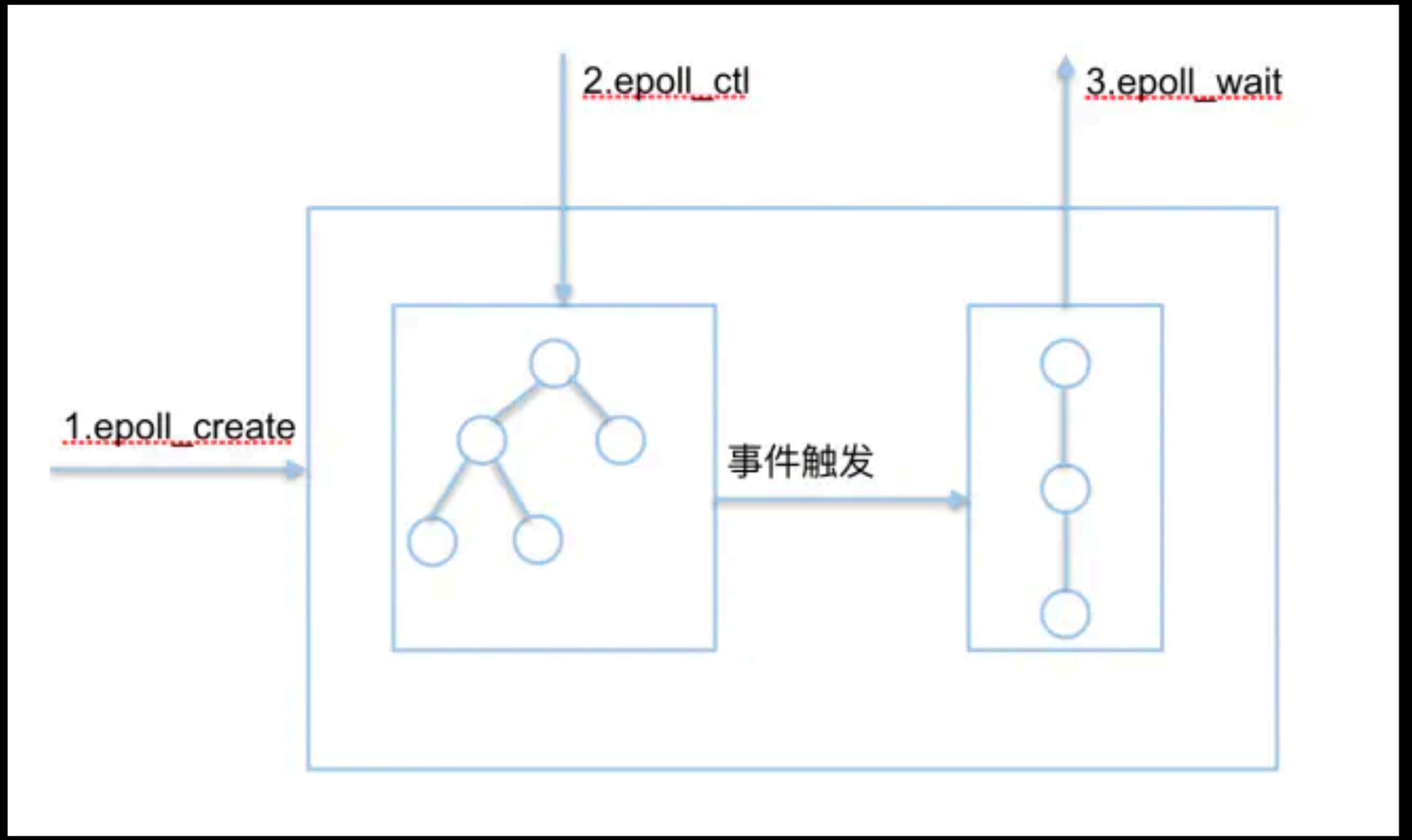
EPOLL_CTL_MOD：修改红黑树中一个描述符



讨论epoll的工作原理是什么

3、int epoll_wait(int epfd， struct epoll_event *events， int maxevents， int timeout);
将被监听的描述符添加到红黑树或从红黑树中删除或者对监听事件进行修改

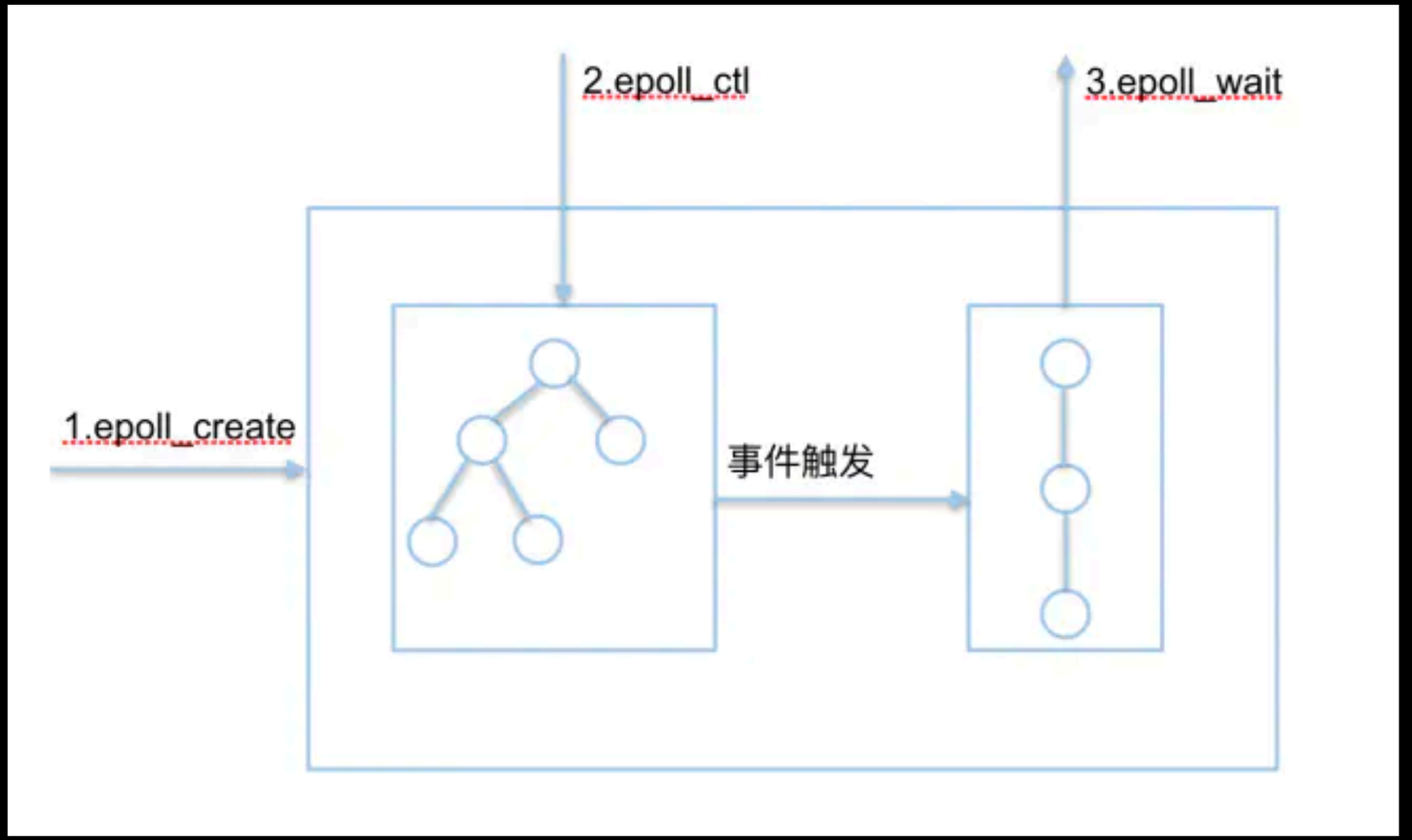
op参数说明操作类型：
EPOLL_CTL_ADD：向红黑树中添加一个需要监视的描述符
EPOLL_CTL_DEL： 从红黑树中删除一个描述符
EPOLL_CTL_MOD: 修改红黑树中一个描述符



讨论epoll的工作原理是什么

3、int epoll_wait(int epfd， struct epoll_event *events， int maxevents， int timeout);
将被监听的描述符添加到红黑树或从红黑树中删除或者对监听事件进行修改

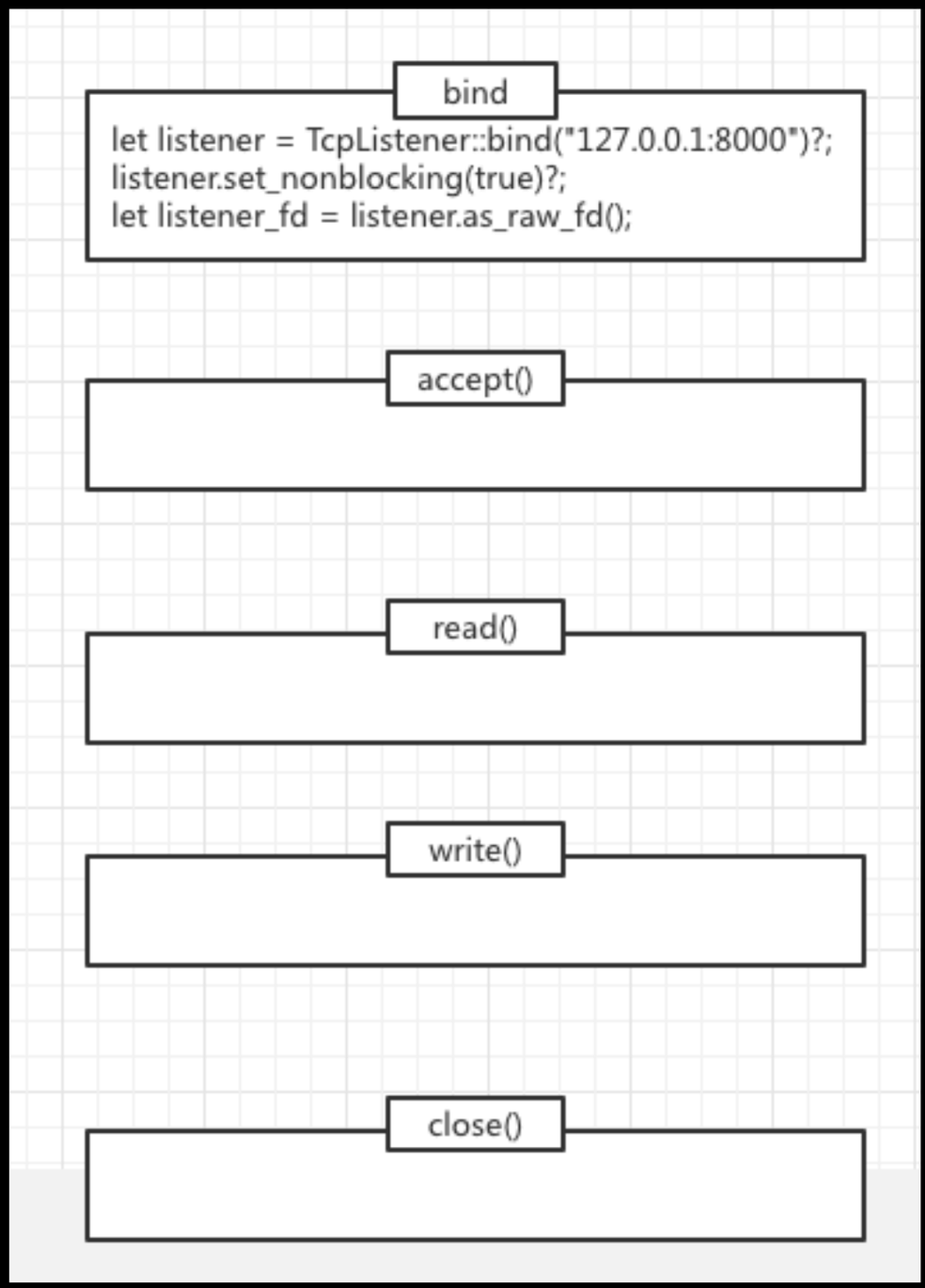
op参数说明操作类型：
EPOLL_CTL_ADD：向红黑树中添加一个需要监视的描述符
EPOLL_CTL_DEL： 从红黑树中删除一个描述符
EPOLL_CTL_MOD: 修改红黑树中一个描述符



rust-epoll-example 代码实践

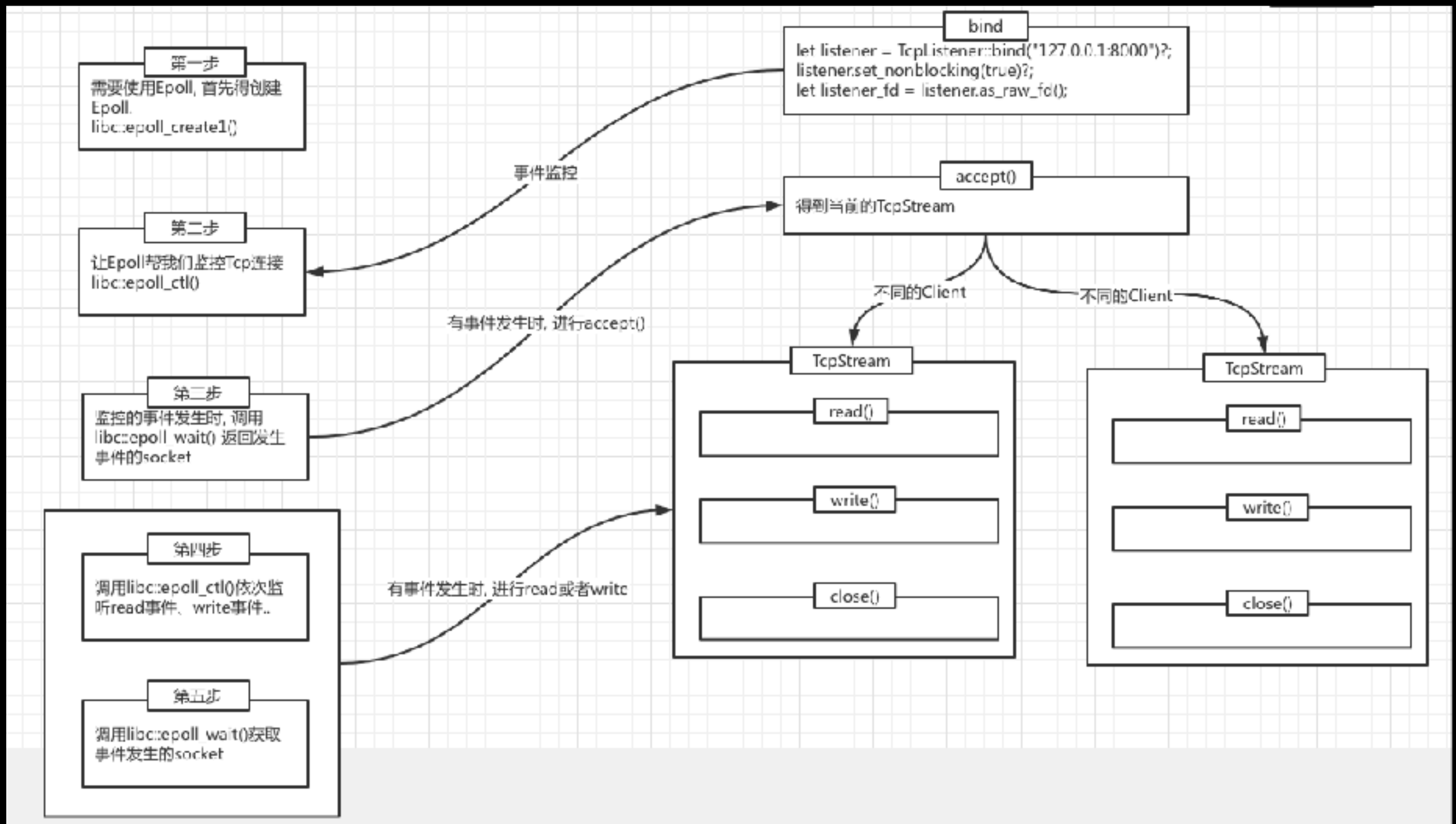
作者也写了一篇文章: https://www.zupzup.org/epoll-with-rust/, 大家可以下来细看.

```rust
use std::net::{TcpListener, TcpStream};
use std::io::{Read, Write};
use std::fs;

fn handle_client(mut stream: TcpStream) {
    let mut buffer = [0; 512];
    stream.read(&mut buffer).unwrap();
    println!("Request: {}", String::from_utf8_lossy(&buffer[..]));

    let get = b"GET / HTTP/1.1\r\n";

    let (status_line, filename) = if buffer.starts_with(get) {
        ("HTTP/1.1 200 OK\r\n\r\n", "main.html")
    } else {
        ("HTTP/1.1 404 NOT FOUND\r\n\r\n", "404.html")
    };

    let contents = fs::read_to_string(filename).unwrap();
    let reponse = format!("{}{}", status_line, contents);

    stream.write(reponse.as_bytes()).unwrap();
    stream.flush().unwrap();
}


fn main() -> std::io::Result<()> {
    let listener = TcpListener::bind("127.0.0.1:8080")?;

    for stream in listener.incoming() {
        handle_client(stream?);
    }
    Ok(())
}
```



rust-epoll-example 代码实践

思考, 引入epoll如何来实现这个功能?



rust-epoll-example 代码实践

讲解事例代码

rust-reactor-executor-example 代码实践

讲解事例代码

examples-minimio 代码实践

讲解事例代码

QA环节

加群一起交流Rust & Datafuse

Datafuse Rust语言中文社区 知数堂 zhishuteng.com