epoll_event

作用

在 epoll_ctl() 和 epoll_wait() 这两个系统调用中均需要使用 epoll_event 这一结构体，该结构体中只有两个成员变量，但是却是 epoll 中最为核心的两个成员变量

```
struct epoll_event {
    uint32_t      events;
    epoll_data_t data;
}
```

```
typedef union epoll_data {
    void     *ptr;
    int       fd;
    uint32_t u32;
    uint64_t u64;
} epoll_data_t
```

其中 events 为位掩码，可以将多个 epoll 事件进行按位或（|）。data 则是系统为用户所预留的一个位置，允许用户自定义一些数据

- events
  - EPOLLIN — 可读事件，用于读取非高优先级的数据
  - EPOLLRDHUP — 对端 socket 关闭，EPOLLRDHUP 是判断对端 socket 是否关闭的首要标志位
  - EPOLLOUT — 可写事件
  - EPOLLET — 采用边缘触发通知，若未指定该 flags，则默认为水平触发通知
  - EPOLLERR — 有错误发生
  - EPOLLHUP — 出现挂断，和 EPOLLERR 看作是有错误发生即可
- epoll_data_t data
  - 从 epoll_data_t 是一个联合结构就可以看出，系统其实也没有给出太多的选择，要么是一个指针，要么就是一个文件描述符。不过，在绝大多数情况下，我们会给该成员变量一个指针，也就是 ptr。
  - int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);
int epoll_wait(int epfd, struct epoll_event *evlist, int maxevents, int timeout);
  - **通过观察这两个系统调用我们可以看到，不管是添加事件，还是获取已就绪的事件，我们都只能通过 epoll_event 完成。也就是说，当我们拿到了 evlist 之后，epoll_event.data 字段是唯一可获知同这个事件相关的文件描述符号的途径**
  - **因此，当我们调用 epoll_ctl() 的时候，要么使用 event.data.fd ，传入一个文件描述符；要么使用 event.data.ptr，传入一个包含该文件描述符的结构体**

通常而言，在一般的 WebServer 开发过程中，我们不仅仅要保存一个 socket fd，还需要保存于该 socket fd 相关的信息，包括 TCP 连接的地址信息，写入数据信息，读取数据信息，等等。因此，我们会将与该 socket fd 以及附加的信息保存在内存中，通过指针进行获取，那么 event.data.ptr 即可传入该指针。Nginx 就是这么干的

边缘触发与水平触发

- 边缘触发 (ET, Edge Trigger)
  - 边缘触发的意思就是当可读/可写事件发生时，内核只会通知一次，后续不再通知
  - 我们以 recv() 为例来描述边缘触发模式
    - 假设我们在读取接收缓冲区数据时，每次就取 2 字节的数据，并且我们假设接收到的任何一个 TCP 包的数据均为 20 字节
    - 当有新的 TCP 包进入接收缓冲区时，内核就会将 epitem 移入双向链表中，并解除 epoll_wait() 的阻塞（如果被阻塞的话），我们一般把这个过程称之为通知
    - **当我们使用 recv() 仅接收 2 字节之后，位于双向链表中的 epitem 将会被移除，直到下一个 TCP 包到达。如果说没有下一个包了，那么我们将永远都不可能获取剩下的 18 字节数据**
  - 所以，在边缘触发模式下，我们在处理可读事件时，必须要尽可能多的读取缓冲区中的数据，直到没有其它数据为止。同时，也正因为内核仅通知一次的特性，使得边缘触发模式的执行效率非常之高
- 水平触发 (LT, Level Trigger)
  - 水平触发的意思就是只要有可读/可写事件发生，并且应用程序没有处理时，那么就会一直在双向链表中
  - 还是上面儿的例子，在水平触发模型下，recv() 读了 2 字节之后，还剩 18 个字节，此时该 TCP 连接依然是可读的。所以下一次的 epoll_wait() 也会返回该 event，应用程序可以继续读取剩下的字节
  - 正因为内核会多次的把同一个事件"重复"地通知，所以水平触发模式又称为慢速模式，效率要比边缘触发模式低。但是胜在稳妥，OS 会为我们兜底