# 疑问

1.socket 创建的描述符和epoll创建的各自用在什么地方？有什么联系？

2.epoll\_ctl函数，将fd的时间改成IN时间之后，会立即出发epoll的读吗？还是代表等待这个描述符的读事件到来呢？

\_\_uint32\_t \_epo\_event = EPOLLIN | EPOLLET | EPOLLONESHOT;

int ret = epoll\_mod(epollfd, fd, static\_cast<void\*>(this), \_epo\_event);

// 修改描述符状态

int epoll\_mod(int epoll\_fd, int fd, void \*request, \_\_uint32\_t events)

{

struct epoll\_event event;

event.data.ptr = request;

event.events = events;

if(epoll\_ctl(epoll\_fd, EPOLL\_CTL\_MOD, fd, &event) < 0)

{

perror("epoll\_mod error");

return -1;

}

return 0;

}

参考：　<https://blog.csdn.net/yusiguyuan/article/details/15027821>　得出应该是注册该事件．

EPOLLIN： 触发该事件，表示对应的文件描述符上有可读数据。(包括对端SOCKET正常关闭)；

EPOLLONESHOT：只监听一次事件，当监听完这次事件之后，如果还需要继续监听这个socket的话，需要再次把这个socket加入到EPOLL队列里。

3.

# 笔记

EventLoop 一旦loop()起来，是个死循环，只有调用析构函数时才会停止．