# epoll:水平触发与边缘触发

**[Linux架构师](https://www.zhihu.com/people/nan-gui-72-90-50" \t "/Users/yangwenshuo/Documents\\x/_blank)**

21 人赞同了该文章

select和poll都只提供了一个函数:select或者poll函数。  
而epoll提供了三个函数，epoll\_create,epoll\_ctl和epoll\_wait，epoll\_create是创建一个epoll句柄；epoll\_ctl是注册要监听的事件类型；epoll\_wait则是等待事件的产生。

epoll除了提供select/poll那种IO事件的水平触发（Level Triggered）外，还提供了边缘触发（Edge Triggered），这就使得用户空间程序有可能缓存IO状态，减少epoll\_wait/epoll\_pwait的调用，提高应用程序效率。

## **水平触发(level-trggered)**

只要文件描述符关联的读内核缓冲区非空，有数据可以读取，就一直发出可读信号进行通知，  
当文件描述符关联的内核写缓冲区不满，有空间可以写入，就一直发出可写信号进行通知  
LT模式支持阻塞和非阻塞两种方式。epoll默认的模式是LT。

## **边缘触发(edge-triggered)**

当文件描述符关联的读内核缓冲区由空转化为非空的时候，则发出可读信号进行通知，  
当文件描述符关联的内核写缓冲区由满转化为不满的时候，则发出可写信号进行通知  
两者的区别在哪里呢？水平触发是只要读缓冲区有数据，就会一直触发可读信号，而边缘触发仅仅在空变为非空的时候通知一次，

LT(level triggered)是缺省的工作方式，并且同时支持block和no-block socket.在这种做法中，内核告诉你一个文件描述符是否就绪了，然后你可以对这个就绪的fd进行IO操作。如果你不作任何操作，内核还是会继续通知你的，所以，这种模式编程出错误可能性要小一点。传统的select/poll都是这种模型的代表．

## **水平触发和边缘触发模式区别**

* 读缓冲区刚开始是空的
* 读缓冲区写入2KB数据
* 水平触发和边缘触发模式此时都会发出可读信号
* 收到信号通知后，读取了1kb的数据，读缓冲区还剩余1KB数据
* 水平触发会再次进行通知，而边缘触发不会再进行通知  
  所以，边缘触发需要一次性的把缓冲区的数据读完为止，也就是一直读，直到读到EGAIN为止，EGAIN说明缓冲区已经空了，因为这一点，边缘触发需要设置文件句柄为非阻塞

//水平触发

ret = read(fd, buf, sizeof(buf));

//边缘触发

while(true) {

ret = read(fd, buf, sizeof(buf);

if (ret == EAGAIN) break;

}

## **设置方法**

int epoll\_ctl(int epfd, int op, int fd, struct epoll\_event \*event);  
epoll的事件注册函数，它不同与select()是在监听事件时告诉内核要监听什么类型的事件，而是在这里先注册要监听的事件类型。

第一个参数是epoll\_create()的返回值，

第二个参数表示动作，用三个宏来表示：  
EPOLL\_CTL\_ADD：注册新的fd到epfd中；  
EPOLL\_CTL\_MOD：修改已经注册的fd的监听事件；  
EPOLL\_CTL\_DEL：从epfd中删除一个fd；

第三个参数是需要监听的fd，第四个参数是告诉内核需要监听什么事，struct epoll\_event结构如下：

struct epoll\_event {

\_\_uint32\_t events; /\* Epoll events /

epoll\_data\_t data; / User data variable \*/

};

events可以是以下几个宏的集合：  
EPOLLIN ：表示对应的文件描述符可以读（包括对端SOCKET正常关闭）；  
EPOLLOUT：表示对应的文件描述符可以写；  
EPOLLPRI：表示对应的文件描述符有紧急的数据可读（这里应该表示有带外数据到来）；  
EPOLLERR：表示对应的文件描述符发生错误；  
EPOLLHUP：表示对应的文件描述符被挂断；  
EPOLLET： 将EPOLL设为边缘触发(Edge Triggered)模式，这是相对于水平触发(Level Triggered)来说的。  
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。  
  
使用示例

使用ET的例子:nginx  
使用LT的例子:redis