1、EPOLL简介

epoll是linux下的一个处理多路I/O复用的机制，基于事件通知，能够高效的处理多个SOCKET连接。

2、基本函数

使用epoll，基本的函数只有三个：   
(1)创建：epoll\_create   
(2)控制：epoll\_ctl   
(3)监听：epoll\_wait

#### 2.1 创建

函数原型：int epoll\_create(int size);

函数打开一个epoll的文件描述符，用于其他epoll函数调用。关闭的时候使用close函数。

#### 2.2 控制

函数原型：int epoll\_ctl(int epfd, int op, int fd, struct epoll\_event \*event);

函数用于 添加/删除/修改 需要监听的SOCKET。   
参数1是epoll\_create 返回的句柄。   
参数2在三个宏定义中选择：

EPOLL\_CTL\_ADD:添加需要监听的SOCKET和事件

EPOLL\_CTL\_MOD:修改监听的事件

EPOLL\_CTL\_DEL:移除监听目标

参数3是操作的目标SOCKET。   
参数4是操作的事件参数：

typedef union epoll\_data {

void \*ptr;

int fd;

\_\_uint32\_t u32;

\_\_uint64\_t u64;

} epoll\_data\_t;

struct epoll\_event {

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

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

};

需要关注的事件：

EPOLLIN：当目标SOCKET有数据到来时触发。

EPOLLOUT：当目标SOCKET可写进触发。

EPOLLERR：出错时触发。

EPOLLHUP：对方SOCKET挂起时触发。

EPOLLET：使用边缘触发（EdgeTriggered）方式。默认是水平触发(LevelTriggered)。

#### 2.3 监听

函数原型：

int epoll\_wait(int epfd, struct epoll\_event \* events, int maxevents, int timeout);

参数2是一个epoll\_event数组，用来保存收到的信息。   
参数3是数组大小。   
参数4是超时时间，单位毫秒。   
返回值是有事件发生的SOCKET数量。

### 3 水平触发 、边缘触发以及EPOLLOUT

边缘触发，就是状态发生改变时才触发。   
水平触发，则是状态存在就一直会触发。

对于EPOLLIN来说，就是水平触发的话，只要socket里有数据可读，还没读完，则每次调用epoll\_wait都会触发。   
而在边缘触发下，如果一次调用没有把数据都读完的话，则下次调用不会触发，要直到下次对方有数据进来时才能继续。而要想一次把数据读完，则必须把socket设置成非阻塞，要不然程序就卡在read的地方了。

对于EPOLLOUT来说，比较难理解。因为读数据是被动的，可以监听很正常，而发数据应该是个主动的过程，如何算作监听呢？

在水平触发下，如果设置了EPOLLOUT，那么只要socket缓冲区足够空间，就会一直触发可写状态。

而边缘触发下，设置了EPOLLOUT，会在连接时触发一次，然后在下次状态由不可写转化为可写时才触发。写缓冲区满了的情况下会出现不可写的状态。

# [EAGAIN、EWOULDBLOCK、EINTR与非阻塞](http://www.cnblogs.com/bastard/archive/2013/04/10/3012724.html)

**EWOULDBLOCK：**用于非阻塞模式，不需要重新读或者写

**EINTR：**指操作被中断唤醒，需要重新读/写  
  
　　在Linux环境下开发经常会碰到很多错误(设置errno)，其中EAGAIN是其中比较常见的一个错误(比如用在非阻塞操作中)。

从字面上来看，是提示再试一次。这个错误经常出现在当应用程序进行一些非阻塞(non-blocking)操作(对文件或socket)的时候。

　　例如，以 O\_NONBLOCK的标志打开文件/socket/FIFO，如果你连续做read操作而没有数据可读。此时程序不会阻塞起来等待数据准备就绪返 回，

read函数会返回一个错误EAGAIN，提示你的应用程序现在没有数据可读请稍后再试。

　　又例如，当一个系统调用(比如fork)因为没有足够的资源(比如虚拟内存)而执行失败，返回EAGAIN提示其再调用一次(也许下次就能成功)。

**EAGAIN：**Linux - 非阻塞socket编程处理EAGAIN错误

　　在linux进行非阻塞的socket接收数据时经常出现Resource temporarily unavailable，errno代码为11(EAGAIN)，这是什么意思？

这表明你在非阻塞模式下调用了阻塞操作，在该操作没有完成就返回这个错误，这个错误不会破坏socket的同步，不用管它，下次循环接着recv就可以。

对非阻塞socket而言，EAGAIN不是一种错误。在VxWorks和Windows上，EAGAIN的名字叫做EWOULDBLOCK。

另外，如果出现EINTR即errno为4，错误描述Interrupted system call，操作也应该继续。

最后，如果recv的返回值为0，那表明连接已经断开，我们的接收操作也应该结束。