poll()函数：这个函数是某些Unix系统提供的用于执行与select()函数同等功能的函数，下面是这个函数的声明：

#include <poll.h>

int poll(struct pollfd fds[], nfds_t nfds, int timeout)；

参数说明：

fds：是一个struct pollfd结构类型的数组，用于存放需要检测其状态的Socket描述符；每当调用这个函数之后，系统不会清空这个数组，操作起来比较方便；特别是对于 socket连接比较多的情况下，在一定程度上可以提高处理的效率；这一点与select()函数不同，调用select()函数之后，select() 函数会清空它所检测的socket描述符集合，导致每次调用select()之前都必须把socket描述符重新加入到待检测的集合中；因 此，select()函数适合于只检测一个socket描述符的情况，而poll()函数适合于大量socket描述符的情况；

nfds：nfds_t类型的参数，用于标记数组fds中的结构体元素的总数量；

timeout：是poll函数调用阻塞的时间，单位：毫秒；

返回值：

>0：数组fds中准备好读、写或出错状态的那些socket描述符的总数量；

==0：数组fds中没有任何socket描述符准备好读、写，或出错；此时poll超时，超时时间是timeout毫秒；换句话说，如果所检测的 socket描述符上没有任何事件发生的话，那么poll()函数会阻塞timeout所指定的毫秒时间长度之后返回，如果timeout==0，那么 poll() 函数立即返回而不阻塞，如果timeout==INFTIM，那么poll() 函数会一直阻塞下去，直到所检测的socket描述符上的感兴趣的事件发 生是才返回，如果感兴趣的事件永远不发生，那么poll()就会永远阻塞下去；

-1： poll函数调用失败，同时会自动设置全局变量errno；

如果待检测的socket描述符为负值，则对这个描述符的检测就会被忽略，也就是不会对成员变量events进行检测，在events上注册的事件也会被忽略，poll()函数返回的时候，会把成员变量revents设置为0，表示没有事件发生；

另外，poll() 函数不会受到socket描述符上的O_NDELAY标记和O_NONBLOCK标记的影响和制约，也就是说，不管socket是阻塞的还是非阻塞 的，poll()函数都不会收到影响；而select()函数则不同，select()函数会受到O_NDELAY标记和O_NONBLOCK标记的影 响，如果socket是阻塞的socket，则调用select()跟不调用select()时的效果是一样的，socket仍然是阻塞式TCP通讯，相 反，如果socket是非阻塞的socket，那么调用select()时就可以实现非阻塞式TCP通讯；

所以poll() 函数的功能和返回值的含义与 select() 函数的功能和返回值的含义是完全一样的，两者之间的差别就是内部实现方式不一样，select()函数基本上可以在所有支持文件描述符操作的系统平台上运 行(如：**Linux** 、Unix 、Windows、MacOS等)，可移植性好，而poll()函数则只有个别的的操作系统提供支持(如：SunOS、Solaris、AIX、HP提供 支持，但是Linux不提供支持)，可移植性差；

strust pollfd结构说明：

```
typedef struct pollfd {
        int fd;                         /* 需要被检测或选择的文件描述符*/
        short events;                   /* 对文件描述符fd上感兴趣的事件 */
        short revents;                  /* 文件描述符fd上当前实际发生的事件*/
} pollfd_t;
typedef unsigned long   nfds_t;
```

经常检测的事件标记： POLLIN/POLLRDNORM(可读)、POLLOUT/POLLWRNORM(可写)、POLLERR(出错)

如果是对一个描述符上的多个事件感兴趣的话，可以把这些常量标记之间进行按位或运算就可以了；

比如：对socket描述符fd上的读、写、异常事件感兴趣，就可以这样做：struct pollfd fds;

fds[nIndex].events=POLLIN | POLLOUT | POLLERR；

当 poll()函数返回时，要判断所检测的socket描述符上发生的事件，可以这样做： struct pollfd fds;

检测可读TCP连接请求：

if((fds[nIndex].revents & POLLIN) == POLLIN){//接收数据/调用accept()接收连接请求}

检测可写：

if((fds[nIndex].revents & POLLOUT) == POLLOUT){//发送数据}

检测异常：

if((fds[nIndex].revents & POLLERR) == POLLERR){//异常处理}