# (1条消息)LINUX下select设置超时 - @奮 鬥@的专栏 - CSDN博客

LINUX设置连接超时方法：

非阻塞 connect：

在一个 TCP 套接字被设置为非阻塞之后调用 connect ，connect 会立即返回 EINPROGRESS 错误，表示连接操作正在进行中，但是仍未完成，与此同时 TCP 三次握手操作会同时进行。在这之后，我们可以通过调用 **select** 来检查这个链接是否建立成功。

在阻塞套接字的一般情况下，connect ()直到客户端对SYN消息的ACK消息到达之前才会返回。使connect()调用具有超时机制的一个方法是让套接字成为非阻塞的套接字体，然后用select()来等待它完成。

```
s = socket(AF_INET, SOCK_STREAM, 0);
//下面获取套接字的标志
if ((flags = fcntl(s, F_GETFL, 0)) < 0) {
   //错误处理
}

//下面设置套接字为非阻塞
if (fcntl(s, F_SETFL, flags | O_NONBLOCK) < 0) {
   //错误处理
}

if ((retcode = connect(s, (struct sockaddr*)&peer, sizeof(peer)) &&
    errno != EINPROGRESS) {
  //因为套接字设为NONBLOCK，通常情况下，连接在connect()返回
  //之前是不会建立的，因此它会返回EINPROGRESS错误，如果返回
  //任何其他错误，则要进行错误处理
}

if (0 == retcode) {  //如果connect()返回0则连接已建立
   //下面恢复套接字阻塞状态
   if (fcntl(s, F_SETFL, flags) < 0) {
      //错误处理
   }

   //下面是连接成功后要执行的代码

       exit(0)
}
```

```
（要设备send/recv超时只需从此处开始修改相应值，前面不用）
FD_ZERO(&rdevents);
FD_SET(s, &rdevents);  //把先前的套接字加到读集合里面
wrevents = rdevents;   //写集合
exevents = rdevents;   //异常集合

tv.tv_sec = 5;  //设置时间为5秒
tv_tv_usec = 0;

retcode = select(s+1, &rdevents, &wrevents, &exevents, &tv);
if (retcode < 0) {  //select返回错误???
   //错误处理
}
else if (0 == retcode) {  //select 超时???
   //超时处理
}
esle {
   //套接字已经准备好
   if (!FD_ISSET(s, &rdevents) && !FD_ISSET(s, &wrevents)) {
      //connect()失败，进行错处理
   }

   if (getsockopt(s, SOL_SOCKET, SO_ERROR, &err, &len) < 0) {
      //getsockopt()失败，进行错处理
   }

   if (err != 0) {
      //connect()失败，进行错处理
   }
（send/recv超时到此为止，返回send()/recv()函数）


   //到这里说明connect()正确返回
   //下面恢复套接字阻塞状态
   if (fcntl(s, F_SETFL, flags) < 0) {
      //错误处理
   }

   //下面是连接成功后要执行的代码
   exit(0)
```

处理非阻塞 connect 的步骤：
第一步，创建 socket，返回套接字描述符；
第二步，调用 fcntl 或 ioctlsocket 把套接口描述符设置成非阻塞；
第三步，调用 connect 开始建立连接；
第四步，判断连接是否成功建立：

A)如果 connect 返回 0 ，表示连接成功(服务器和客户端在同一台机器上时就有可能发生这种情况)；

B)调用 select 来判定连接建立的是否成功；

如果 select 返回 0 ，则表示在 select 的超时时间内未能成功建立连接；我们需要返回超时错误给用户，同时关闭连接，以防止TCP三次握手继续进行下去；

如果 select 返回大于 0 的值，则说明检测到可读或可写或异常的套接字描述符存在；此时我们可以通过调用 getsockopt 来检测集合中的套接口上是否存在待处理的错误，如果连接建立是成功的，则通过 getsockopt(sockfd,SOL_SOCKET,SO_ERROR,(char *)&error,&len) 获取的 error 值将是 0 ，如果建立连接时遇到错误，则 error 的值是连接错误所对应的 errno 值，比如ECONNREFUSED，ETIMEDOUT等。