**http://www.cnblogs.com/hjslovewcl/archive/2011/03/16/2314330.html**

**一.Select 函数详细介绍**

    Select在Socket编程中还是比较重要的，可是对于初学Socket的人来说都不太爱用Select写程序，他们只是习惯写诸如connect、 accept、recv或recvfrom这样的阻塞程序（所谓阻塞方式block，顾名思义，就是进程或是线程执行到这些函数时必须等待某个事件的发 生，如果事件没有发生，进程或线程就被阻塞，函数不能立即返回）。   
    可是使用Select就可以完成非阻塞（所谓非阻塞方式non-block，就是进程或线程执行此函数时不必非要等待事件的发生，一旦执行肯定返回，以返 回值的不同来反映函数的执行情况，如果事件发生则与阻塞方式相同，若事件没有发生则返回一个代码来告知事件未发生，而进程或线程继续执行，所以效率较高） 方式工作的程序，它能够监视我们需要监视的文件描述符的变化情况--读写或是异常。   
  
下面详细介绍一下：   
Select的函数格式(我所说的是Unix系统下的伯克利socket编程，和windows下的有区别，一会儿说明)：   
**int select(int maxfdp,fd\_set \*readfds,fd\_set \*writefds,fd\_set \*errorfds,struct timeval \*timeout);**先说明两个结构体：   
**第一** .struct fd\_set可以理解为一个集合，这个集合中存放的是文件描述符(filedescriptor)，即文件句柄，这可以是我们所说的普通意义的文件，当然 Unix下任何设备、管道、FIFO等都是文件形式，全部包括在内，所以毫无疑问一个socket就是一个文件，socket句柄就是一个文件描述符。   
fd\_set集合可以通过一些宏由人为来操作，比如：  
FD\_ZERO(fd\_set \*); 清空集合  
FD\_SET(int, fd\_set \*); 将一个给定的文件描述符加入集合之中  
FD\_CLR(int, fd\_set\*); 将一个给定的文件描述符从集合中删除  
检查集合中指定的文件描述符是否可以读写FD\_ISSET(int ,fd\_set\* )。一会儿举例说明。   
  
**第二** .struct timeval 是一个大家常用的结构，用来代表时间值，有两个成员，一个是秒数，另一个是毫秒。 具体解释select的参数：   
int maxfdp：是一个整数值，是指集合中所有文件描述符的范围，即所有文件描述符的最大值加1，不能错！在Windows中这个参数的值无所谓，可以设置不正确。   
  
fd\_set\* readfds：是指向fd\_set结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的读变化的，即我们关心是否可以从这些文件中 读取数据了，如果这个集合中有一个文件可读，select就会返回一个大于0的值，表示有文件可读，如果没有可读的文件，则根据timeout参数再判断 是否超时，若超出timeout的时间，select返回0，若发生错误返回负值。可以传入NULL值，表示不关心任何文件的读变化。   
fd\_set\* writefds：是指向fd\_set结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的写变化的，即我们关心是否可以向这些文件 中写入数据了，如果这个集合中有一个文件可写，select就会返回一个大于0的值，表示有文件可写，如果没有可写的文件，则根据timeout参数再判 断是否超时，若超出timeout的时间，select返回0，若发生错误返回负值。可以传入NULL值，表示不关心任何文件的写变化。   
fd\_set \* errorfds：同上面两个参数的意图，用来监视文件错误异常。   
  
struct timeval\* timeout：是select的超时时间，这个参数至关重要，它可以使select处于三种状态：  
第一，若将NULL以形参传入，即不传入时间结构，就是将select置于阻塞状态，一定等到监视文件描述符集合中某个文件描述符发生变化为止；  
第二，若将时间值设为0秒0毫秒，就变成一个纯粹的非阻塞函数，不管文件描述符是否有变化，都立刻返回继续执行，文件无变化返回0，有变化返回一个正值；  
第三，timeout的值大于0，这就是等待的超时时间，即select在timeout时间内阻塞，超时时间之内有事件到来就返回了，否则在超时后不管怎样一定返回，返回值同上述。   
  
**第三.select 返回值：**   
负值：select错误   
正值：某些文件可读写或出错   
0：等待超时，没有可读写或错误的文件   
在有了select后可以写出像样的网络程序来！  
举个简单的例子，就是从网络上接受数据写入一个文件中。

1. main()
2. {
3. int sock;
4. FILE \*fp;
5. **struct** fd\_set fds;
6. **struct** timeval timeout={3,0}; //select等待3秒，3秒轮询，要非阻塞就置0
7. char buffer[256]={0}; //256字节的接收缓冲区
8. /\* 假定已经建立UDP连接，具体过程不写，简单，当然TCP也同理，主机ip和port都已经给定，要写的文件已经打开
9. sock=socket(...);
10. bind(...);
11. fp=fopen(...); \*/
12. **while**(1)
13. {
14. FD\_ZERO(&fds); //每次循环都要清空集合，否则不能检测描述符变化
15. FD\_SET(sock,&fds); //添加描述符
16. FD\_SET(fp,&fds); //同上
17. maxfdp=sock>fp?sock+1:fp+1;    //描述符最大值加1
18. **switch**(select(maxfdp,&fds,&fds,NULL,&timeout))   //select使用
19. {
20. **case** -1: exit(-1);**break**; //select错误，退出程序
21. **case** 0:**break**; //再次轮询
22. **default**:
23. **if**(FD\_ISSET(sock,&fds)) //测试sock是否可读，即是否网络上有数据
24. {
25. recvfrom(sock,buffer,256,.....);//接受网络数据
26. **if**(FD\_ISSET(fp,&fds)) //测试文件是否可写
27. fwrite(fp,buffer...);//写入文件
28. buffer清空;
29. }// end if break;
30. }// end switch
31. }//end while
32. }//end main

**二.另一个例子：**   
#include<sys/time.h>   
#include<sys/types.h>   
#include<unistd.h>   
定义函数 int select(int n,fd\_set \* readfds,fd\_set \* writefds,fd\_set \* exceptfds,struct timeval \* timeout);   
函数说明 select()用来等待文件描述词状态的改变。参数n代表最大的文件描述词加1，参数readfds、writefds 和exceptfds 称为描述词组，是用来回传该描述词的读，写或例外的状况。底下的宏提供了处理这三种描述词组的方式:

1. FD\_CLR(inr fd,fd\_set\* set)；用来清除描述词组set中相关fd 的位
2. FD\_ISSET(int fd,fd\_set \*set)；用来测试描述词组set中相关fd 的位是否为真
3. FD\_SET（int fd,fd\_set\*set）；用来设置描述词组set中相关fd的位
4. FD\_ZERO（fd\_set \*set）； 用来清除描述词组set的全部位
5. 参数 timeout为结构timeval，用来设置select()的等待时间，其结构定义如下
6. **struct** timeval
7. {
8. time\_t tv\_sec;
9. time\_t tv\_usec;
10. };

返回值 如果参数timeout设为NULL则表示select（）没有timeout。   
错误代码 执行成功则返回文件描述词状态已改变的个数，如果返回0代表在描述词状态改变前已超过timeout时间，当有错误发生时则返回-1，错误原因存于errno，此时参数readfds，writefds，exceptfds和timeout的值变成不可预测。   
EBADF 文件描述词为无效的或该文件已关闭   
EINTR 此调用被信号所中断   
EINVAL 参数n 为负值。   
ENOMEM 核心内存不足   
范例 常见的程序片段:fs\_set readset；   
FD\_ZERO(&readset);   
FD\_SET(fd,&readset);   
select(fd+1,&readset,NULL,NULL,NULL);   
if(FD\_ISSET(fd,readset){……}

下面是linux环境下select的一个简单用法

1. #i nclude <sys/time.h>
2. #i nclude <stdio.h>
3. #i nclude <sys/types.h>
4. #i nclude <sys/stat.h>
5. #i nclude <fcntl.h>
6. #i nclude <assert.h>
7. int main ()
8. {
9. int keyboard;
10. int ret,i;
11. char c;
12. fd\_set readfd;
13. **struct** timeval timeout;
14. keyboard = open("/dev/tty",O\_RDONLY | O\_NONBLOCK);
15. assert(keyboard>0);
16. **while**(1)
17. {
18. timeout.tv\_sec=1;
19. timeout.tv\_usec=0;
20. FD\_ZERO(&readfd);
21. FD\_SET(keyboard,&readfd);
22. ret=select(keyboard+1,&readfd,NULL,NULL,&timeout);
23. **if**(FD\_ISSET(keyboard,&readfd))
24. {
25. i=read(keyboard,&c,1);
26. **if**('/n'==c)
27. **continue**;
28. printf("hehethe input is %c/n",c);
30. **if** ('q'==c)
31. **break**;
32. }
33. }
34. }

用来循环读取键盘输入

将例子程序作一修改，加上了time out,并且考虑了select得所有的情况：

1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <sys/stat.h>
4. #include <fcntl.h>
5. #include <assert.h>
6. int main ()
7. {
8. int keyboard;
9. int ret,i;
10. char c;
11. fd\_set readfd;
12. **struct** timeval timeout;
13. keyboard = open("/dev/tty",O\_RDONLY | O\_NONBLOCK);
14. assert(keyboard>0);
15. **while**(1)
16. {
17. timeout.tv\_sec=5;
18. timeout.tv\_usec=0;
19. FD\_ZERO(&readfd);
20. FD\_SET(keyboard,&readfd);
21. ret=select(keyboard+1,&readfd,NULL,NULL,&timeout);
22. //select error when ret = -1
23. **if** (ret == -1)
24. perror("select error");
25. //data coming when ret>0
26. **else** **if** (ret)
27. {
28. **if**(FD\_ISSET(keyboard,&readfd))
29. {
30. i=read(keyboard,&c,1);
31. **if**('/n'==c)
32. **continue**;
33. printf("hehethe input is %c/n",c);
34. **if** ('q'==c)
35. **break**;
36. }
37. }
38. //time out when ret = 0
39. **else** **if** (ret == 0)
40. printf("time out/n");
41. }
42. }