socket编程 - 非阻塞方式

01

1-1 server在accept后再设为非阻塞，client在connect后再设为非阻塞，连接成功后双方一起进入recv

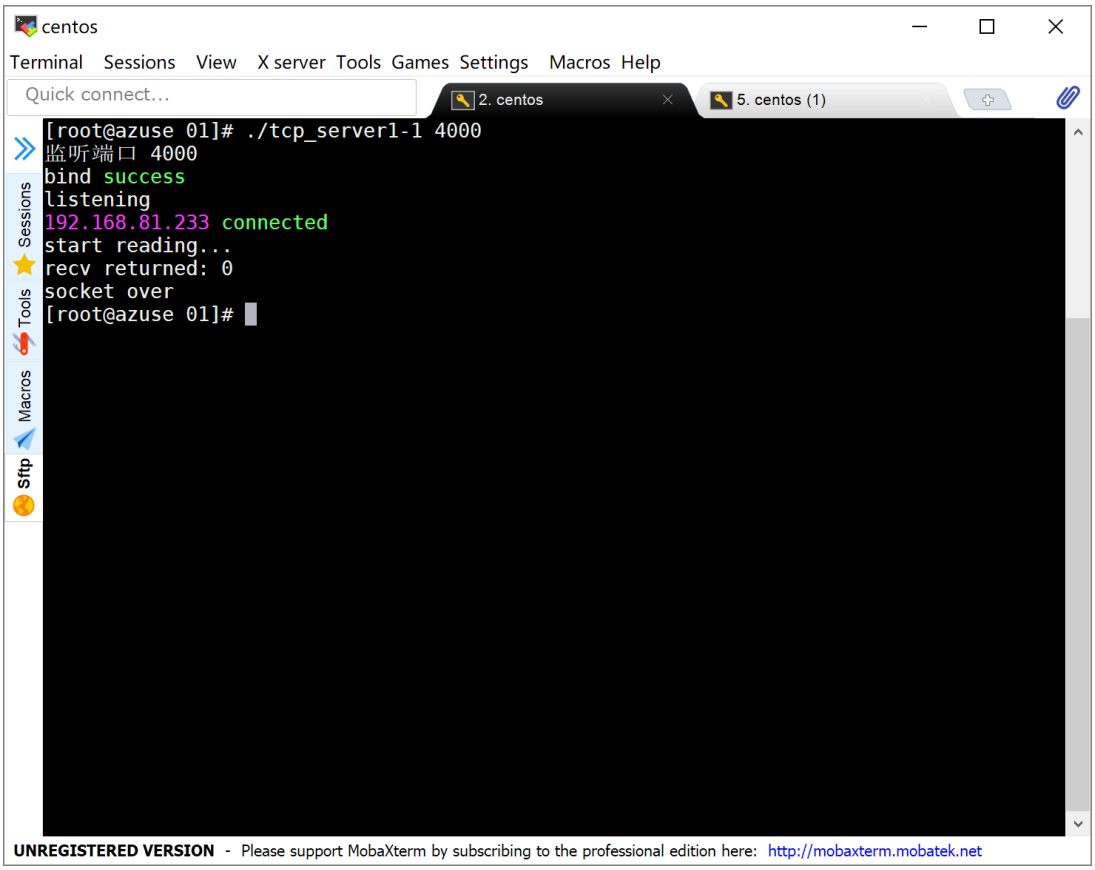
在连接成功后先设置为非阻塞：

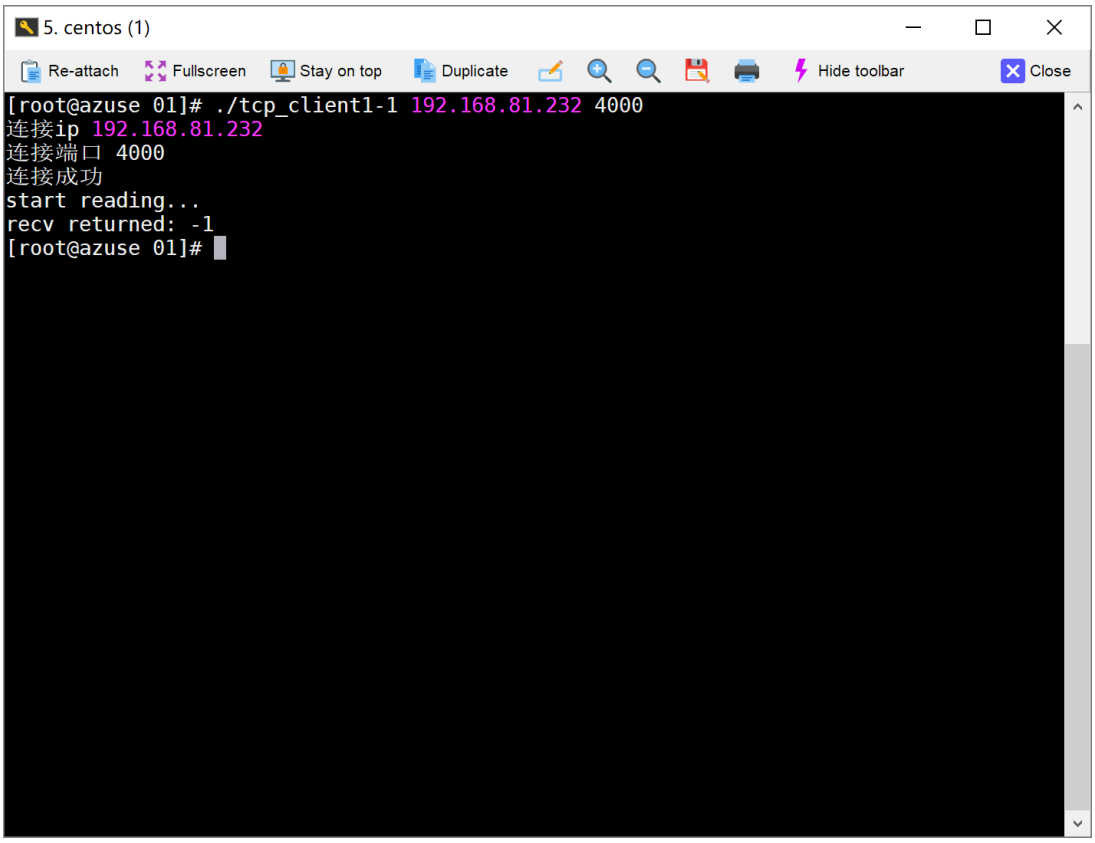
```
int flags = fcntl(server_fd, F_GETFL, 0);
fcntl(server_fd, F_SETFL, flags|O_NONBLOCK);
```

然后进入recv：

```
printf("start reading...\n");
int read = recv(new_socket, buffer, sizeof(buffer), 0);
printf("recv returned: %d\n", read);
```

server端的recv函数返回0，client的recv函数返回-1，程序立即结束没有停在recv上

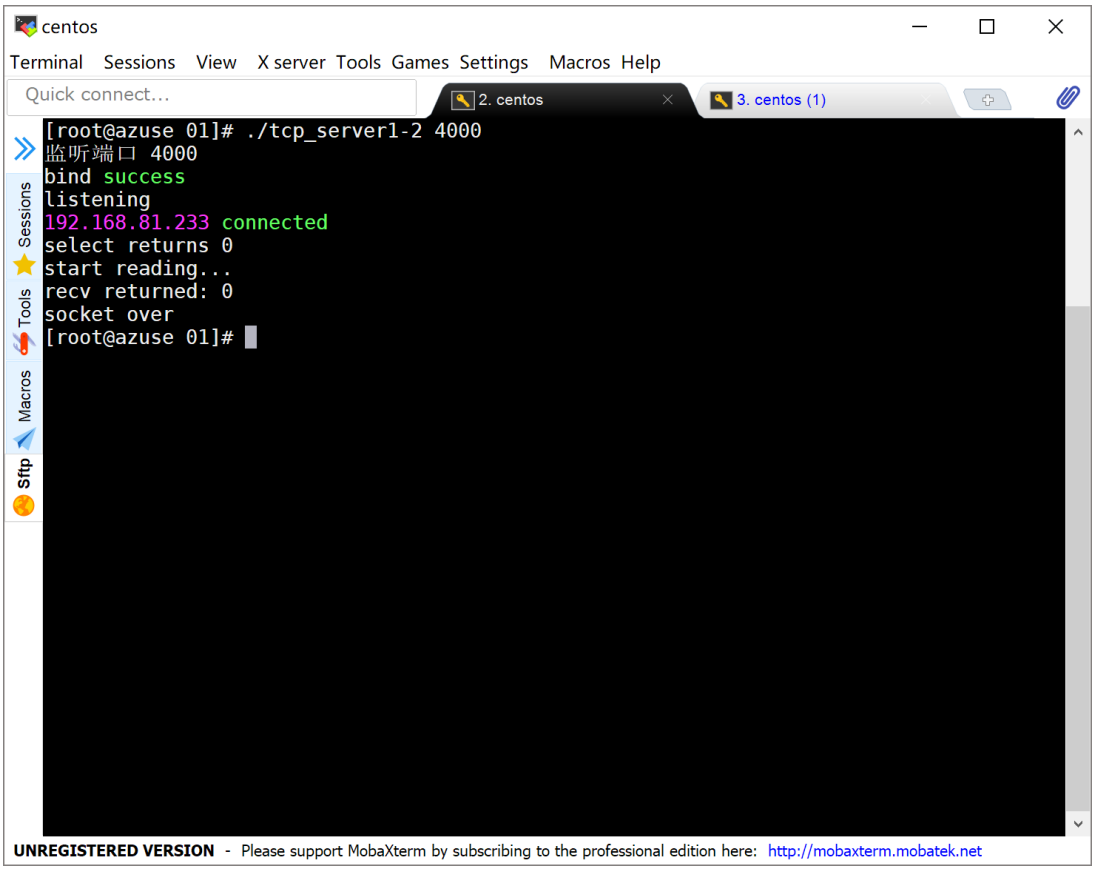


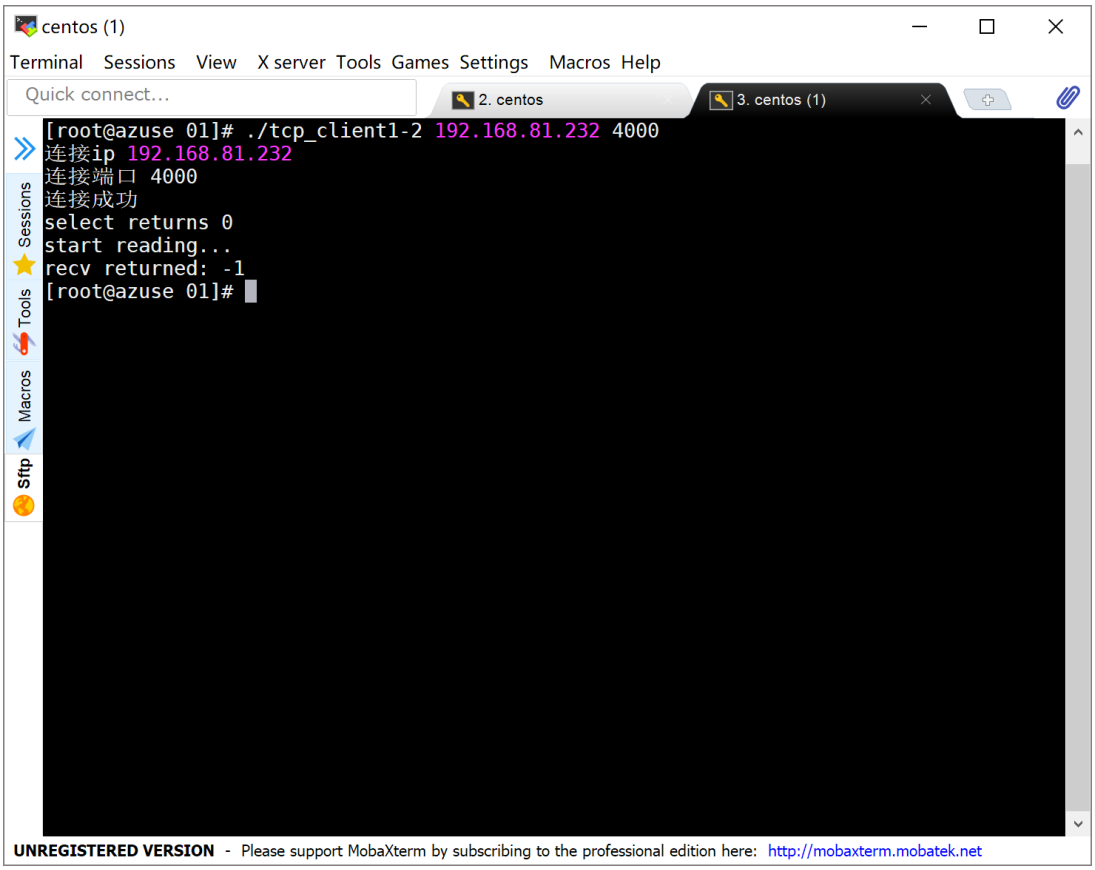


1-2 用select使recv停下来而不立即返回

用一个timeval的tv_sec, tv_usec给select设置超时，select会检测socket是否可读直到超时为止，因为sever与client连接后进入recv状态所以都会停在select，直至超时，server端返回0，client端recv返回-1（如果使用FD_ISSET，则检测到不可读跳过recv）。这里设置select超时为10s。recv后再用一个select阻塞recv，使recv停下来不立即返回

```
        fd_set rfds, wfds;
        struct timeval tv;
        FD_ZERO(&rfds);
        FD_SET(new_socket, &rfds);
        /* set select() time out */
        tv.tv_sec = 10;
        tv.tv_usec = 0;
        int selres = select(new_socket + 1, &rfds, NULL, NULL, &tv);
        printf("select returns %d\n", selres);
        printf("start reading...\n");
        int read = recv(new_socket, buffer, sizeof(buffer), 0);
        printf("recv returned: %d\n", read);
        selres = select(new_socket + 1, NULL, NULL, NULL, &tv);
```





1-3 server与client在socket建立成功后即设置非阻塞

因为accept与connect都变成非阻塞的了，所以需要用select来确保连接已经建立。服务端在accept前select监听端口，客户端在connect后select自己。 server端：

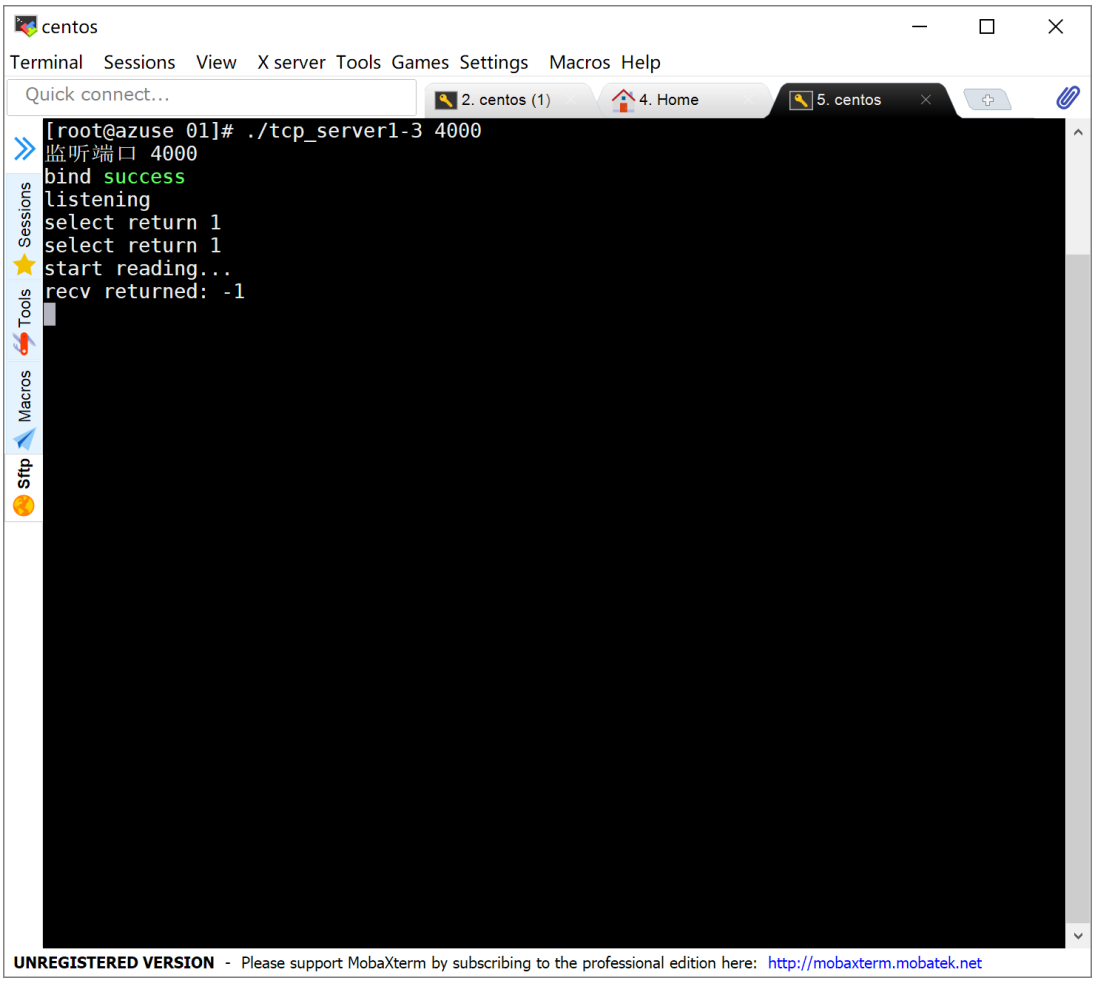
```
fd_set rfds, wfds;
        struct timeval tv;
        FD_ZERO(&rfds);
        FD_ZERO(&wfds);
        FD_SET(server_fd, &rfds);
        FD_SET(server_fd, &wfds);
        /* set select() time out */
        tv.tv_sec = 10;
        tv.tv_usec = 0;

        int selres = select(server_fd + 1, &rfds, &wfds, NULL, NULL);
        printf("select return %d\n", selres);

        new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t
*)&addrlen);
        if(new_socket == -1)
        {
                perror("acccpet failed");
                exit(-1);
        }


        switch (selres)
        {
        case -1:
                printf("select error\n");
                break;
        case 0:
                printf("select time out\n");
                break;
        default:
                printf("select return %d\n", selres);
                flags = fcntl(new_socket, F_GETFL, 0);
                fcntl(new_socket, F_SETFL, flags | O_NONBLOCK);

                printf("start reading...\n");
                int read = recv(new_socket, buffer, sizeof(buffer), 0);
                printf("recv returned: %d\n", read);

                fd_set rfds2, wfds2;
                struct timeval tv;
                FD_ZERO(&rfds2);
                FD_ZERO(&wfds2);
                FD_SET(new_socket, &rfds2);
                FD_SET(new_socket, &wfds2);
                /* set select() time out */
                tv.tv_sec = 10;
                tv.tv_usec = 0;
                selres = select(new_socket + 1, NULL, NULL, NULL, &tv);
                printf("socket over\n");
        }
```

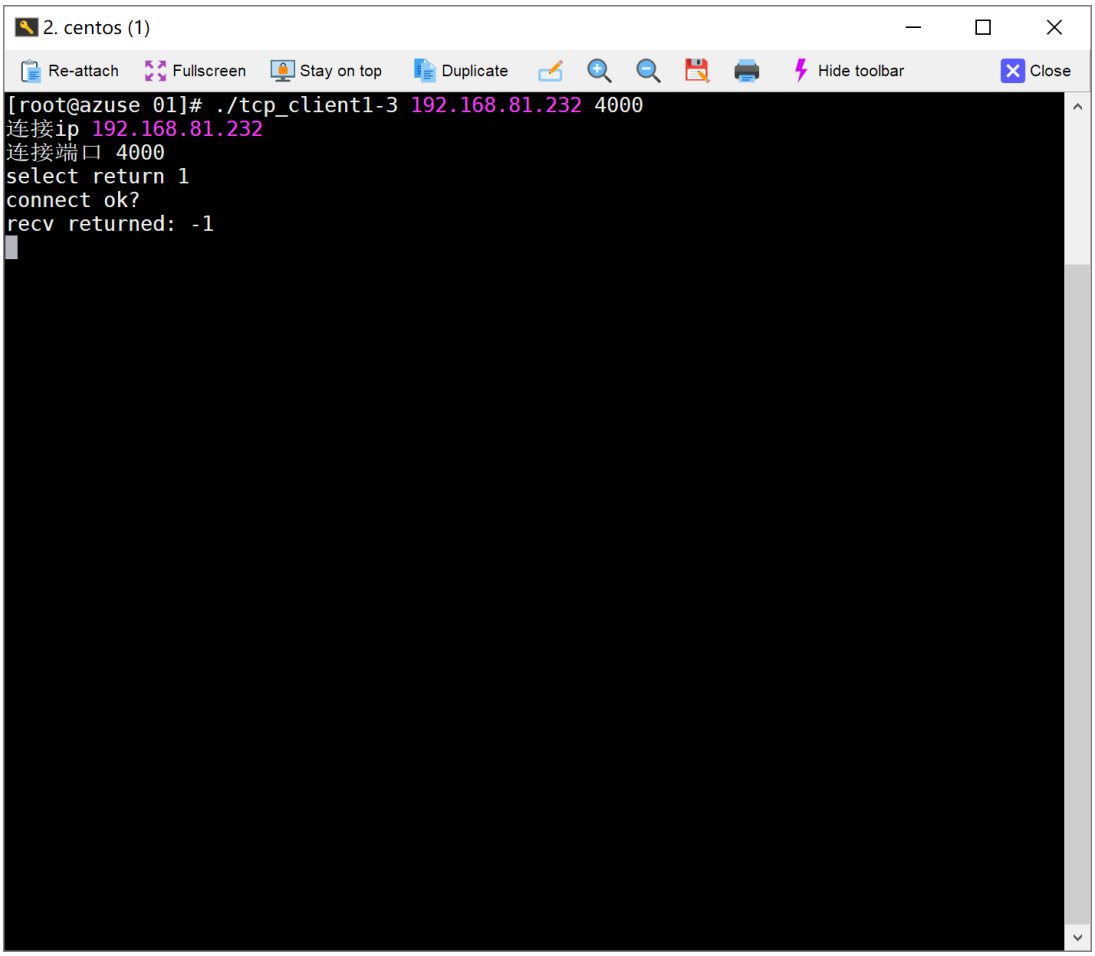
client:

```
    fd_set rfds, wfds;
    struct timeval tv;

    FD_ZERO(&rfds);
    FD_ZERO(&wfds);
    FD_SET(sock, &rfds);
    FD_SET(sock, &wfds);
    /* set select() time out */
    tv.tv_sec = 10;
    tv.tv_usec = 0;

    int res = connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    if (0 == res)
    {
        printf("connect success immediately\n");
    }
    else
    {
        if (errno == EINPROGRESS)
        {
            int selres = select(sock + 1, &rfds, &wfds, NULL, &tv);

            switch (selres)
            {
            case -1:
                printf("select error\n");
                break;
            case 0:
                printf("select time out\n");
                break;
            default:
                printf("select return %d\n", selres);
                if (FD_ISSET(sock, &rfds) && FD_ISSET(sock, &wfds))
                {
                    perror("connect error\n");
                }
                if (FD_ISSET(sock, &rfds) || FD_ISSET(sock, &wfds))
                {

                    int errinfo;
                    socklen_t errlen = sizeof(int);
                    if (-1 == getsockopt(sock, SOL_SOCKET, SO_ERROR, &errinfo,
&errlen))
                    {
                        printf("getsockopt return -1.\n");
                        perror("getsocketopt error:");
                        break;
                    }
                    else if (0 != errinfo)
                    {
                        printf("getsockopt return errinfo = %d.\n", errinfo);
```

```
                        break;
                    }

                    printf("connect ok?\n");

                    int read = recv(sock, buffer, sizeof(buffer), 0);
                    printf("recv returned: %d\n", read);
                                        selres = select(sock + 1, NULL, NULL,
NULL, &tv);

                    printf("socket over\n");
                }
            }
        }

        else
        {
            printf("connect error\n");
        }
    }
```

运行结果:

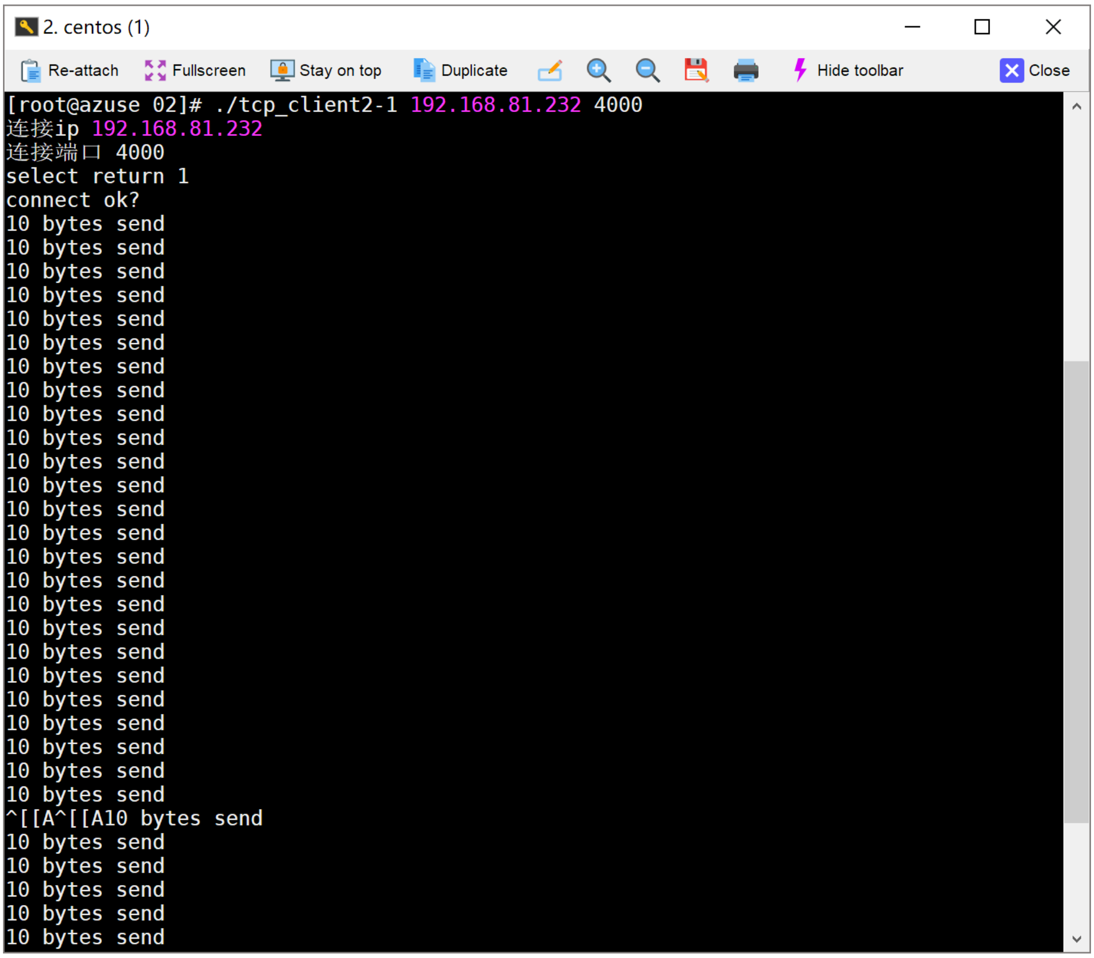

centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos (1)
4. Home
5. centos
Sessions
Tools
Macros
Sftp
[root@azuse 01]# ./tcp_server1-3 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
recv returned: -1
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net


2. centos (1)
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
[root@azuse 01]# ./tcp_client1-3 192.168.81.232 4000
连接ip 192.168.81.232
连接端口 4000
select return 1
connect ok?
recv returned: -1

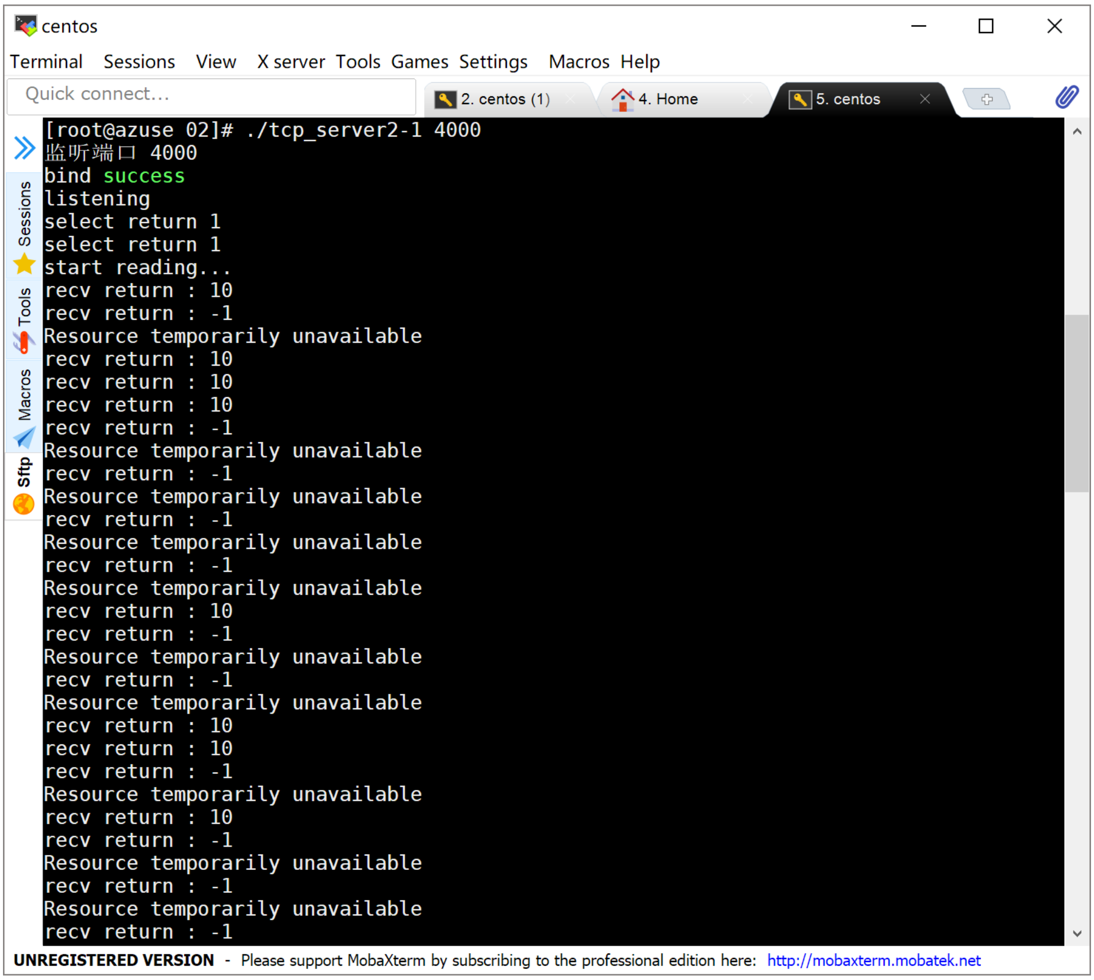
02

2-1 tcp2-1 clientm每秒向server发10字节

正常发送:



```
[root@azuse 02]# ./tcp_client2-1 192.168.81.232 4000
连接ip 192.168.81.232
连接端口 4000
select return 1
connect ok?
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
^[[A^[[A10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
10 bytes send
```



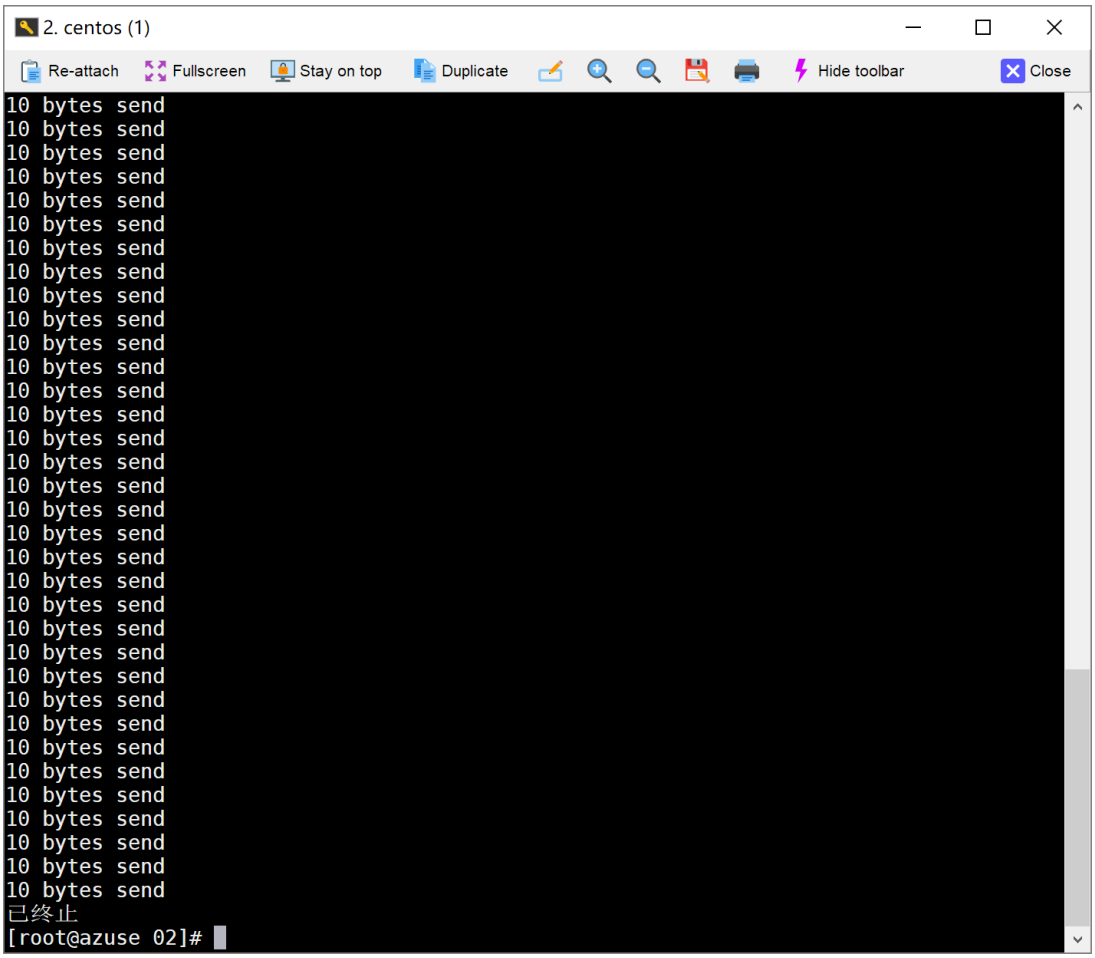
```
[root@azuse 02]# ./tcp_server2-1 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : 10
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
```

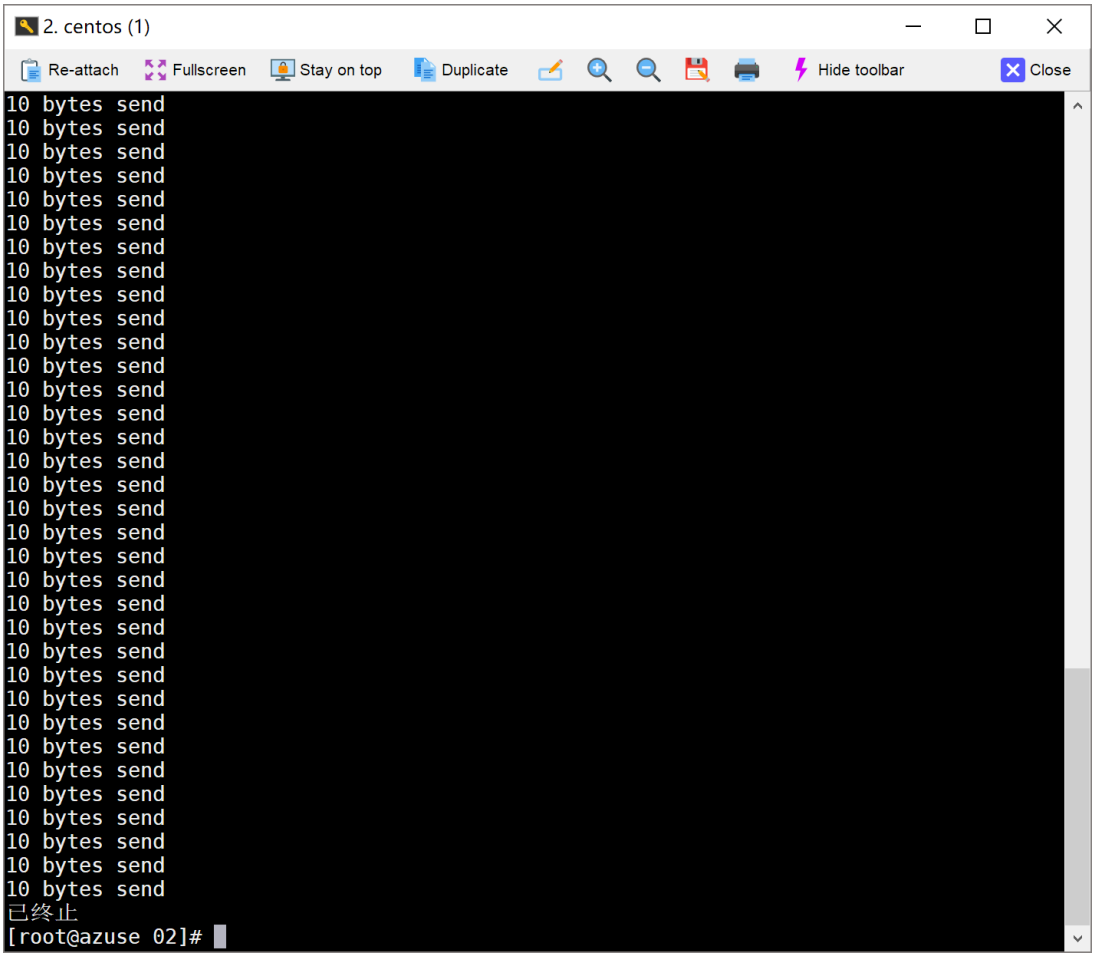
向client发送ctrl+c，server端能检测到中断。

```
[root@azuse 02]# ./tcp_server2-1 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : 10
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
```

```
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : 10
recv return : 10
recv return : 10
Connect closed: Resource temporarily unavailable
socket over
[root@azuse 02]#
```

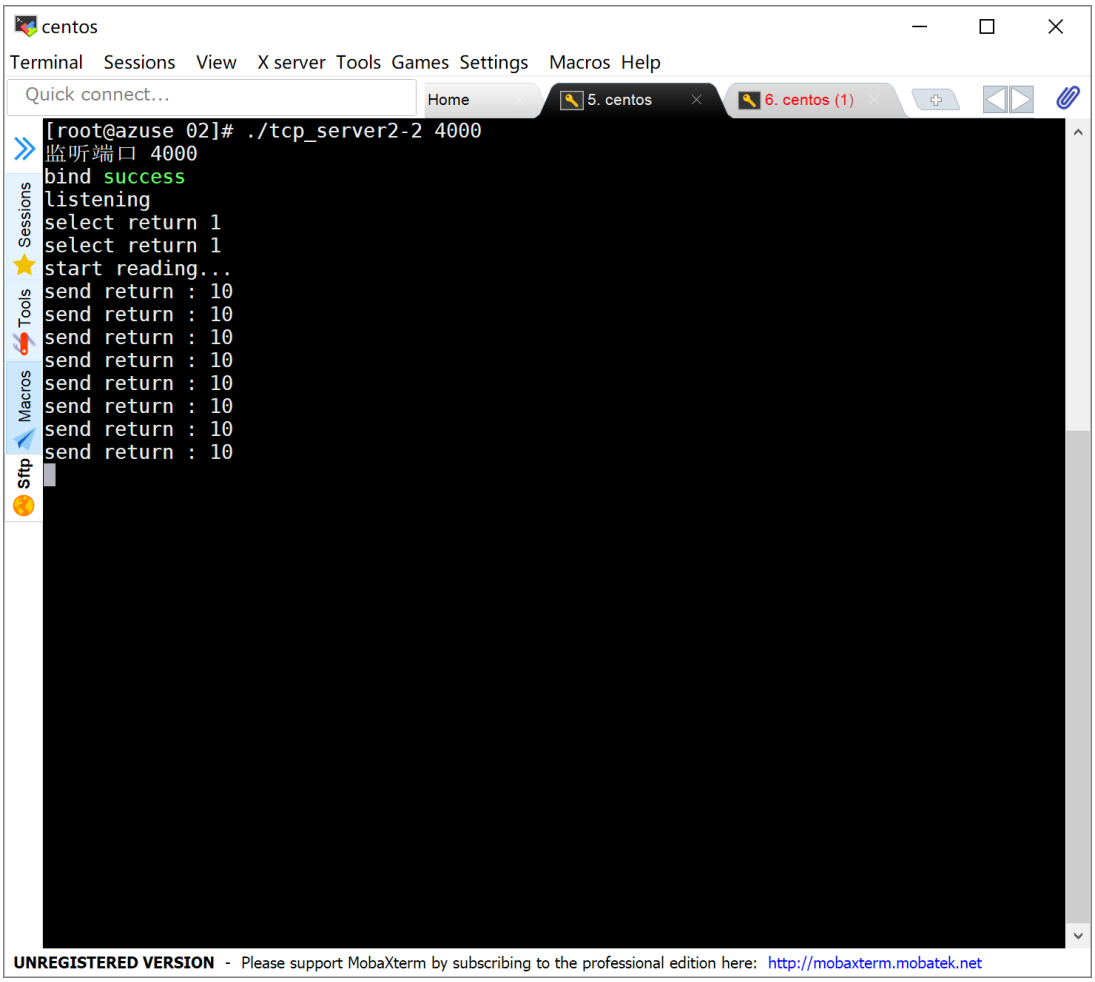
新开一个窗口用kill-9杀死client，server也能检测到中断



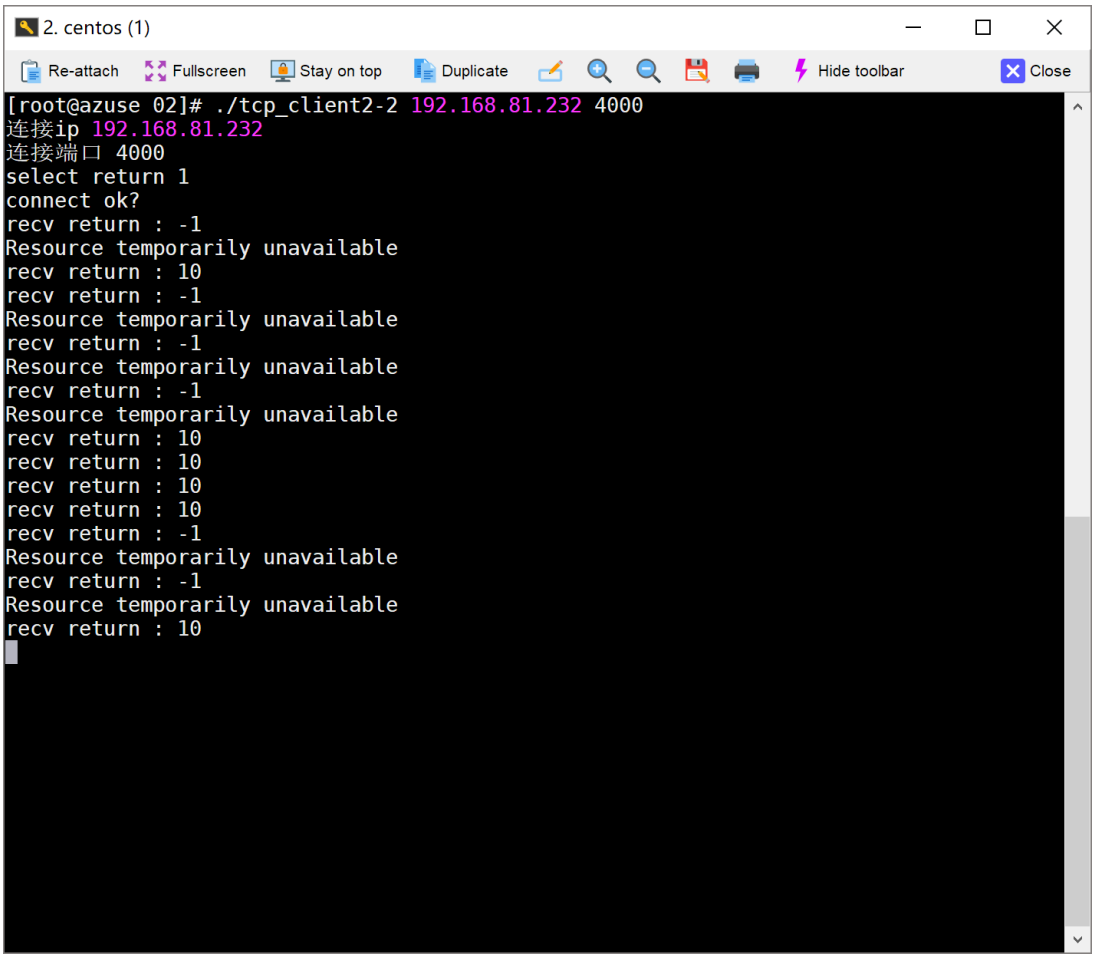


tcp2-2 server发数据，client接收

server正常发送:



client正常接收:



向client发送ctrl+c，server端socket程序收到SIGPIPE退出，用一个信号处理函数输出结果

```
static void sig_sigpipe(int signo){
        printf("error SIGPIPE, exiting\n");
        exit(-1);

}
```

2. centos (1)

```
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
已终止
[root@azuse 02]#
```

centos

```
[root@azuse 02]# ./tcp_server2-2 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
send return : 10
error SIGPIPE, exiting
[root@azuse 02]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

用一个新开的窗口kill -9 client，server端收到SIGPIPE退出，用一个信号处理函数输出结果

2. centos (1)

```
[root@azuse 02]# ./tcp_client2-2 192.168.81.232 4000
连接ip 192.168.81.232
连接端口 4000
select return 1
connect ok?
recv return : -1
Resource temporarily unavailable
recv return : 10
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : -1
Resource temporarily unavailable
recv return : 10
^C
[root@azuse 02]#
```

centos

```
[root@azuse 02]# ./tcp_server2-2 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
send return : 10
send return : 10
send return : 10
error SIGPIPE, exit
[root@azuse 02]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

03

tcp3-1 server端并发发送数据，每次10字节，间隔1s，并同时用大小100的缓冲区收数据，client发数据每次15字节间隔3s，并同时用大小100的缓冲区收数据。

使用一个全局写flag，设置client的alarm每3s响一次，将write_flag至1，然后再main的死循环中send并把write_flag至0，server同，alarm间隔为1s。

```
[root@azuse 03]# ./tcp_server3-1 4000
监听端口 4000
bind success
listening
select return 1
select return 1
start reading...
10 bytes send
15 bytes read
10 bytes send
10 bytes send
15 bytes read
10 bytes send
10 bytes send
10 bytes send
15 bytes read
10 bytes send
10 bytes send
10 bytes send
```

```
[root@azuse 03]# ./tcp_client3-1 192.168.81.232 4000
连接ip 192.168.81.232
连接端口 4000
select return 1
connect ok?
15 bytes send
10 bytes read
10 bytes read
10 bytes read
15 bytes send
10 bytes read
10 bytes read
10 bytes read
15 bytes send
10 bytes read
10 bytes read
10 bytes read
15 bytes send
10 bytes read
10 bytes read
```

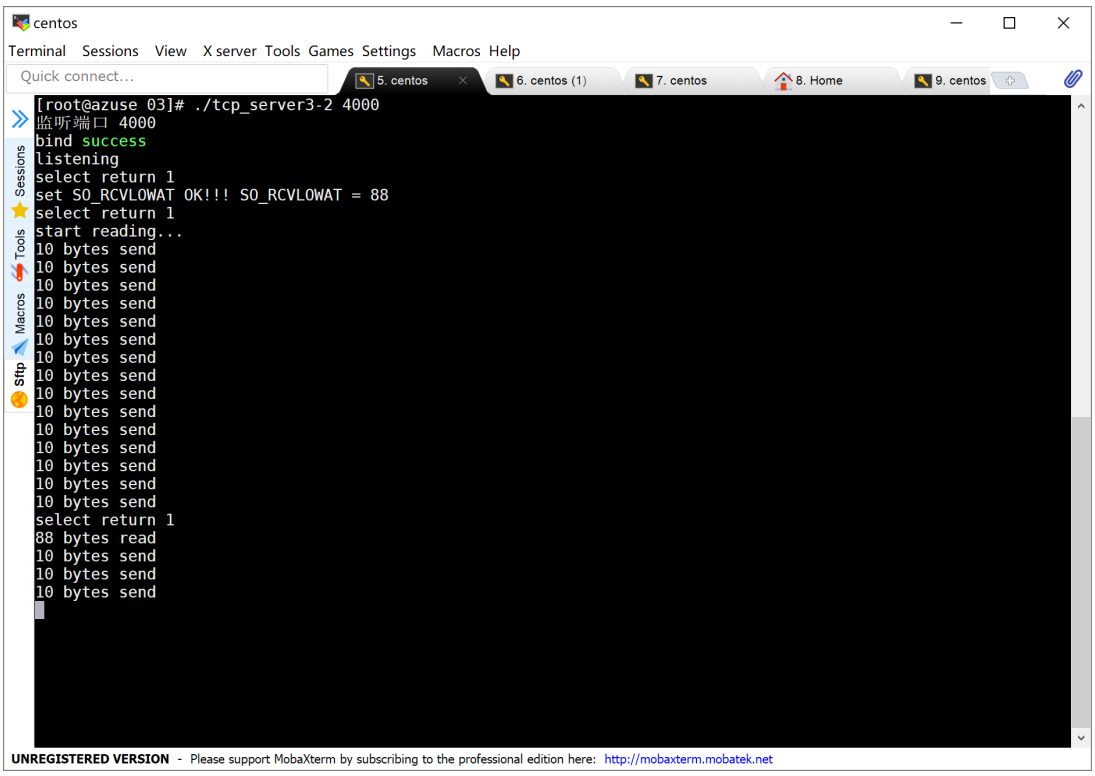
tcp3-2 recv 不读满88字节不返回

用setsockopt将SO_RCVLOWAT设置为88，然后再read前select rfds等系统通知可读后再一次读取88字节

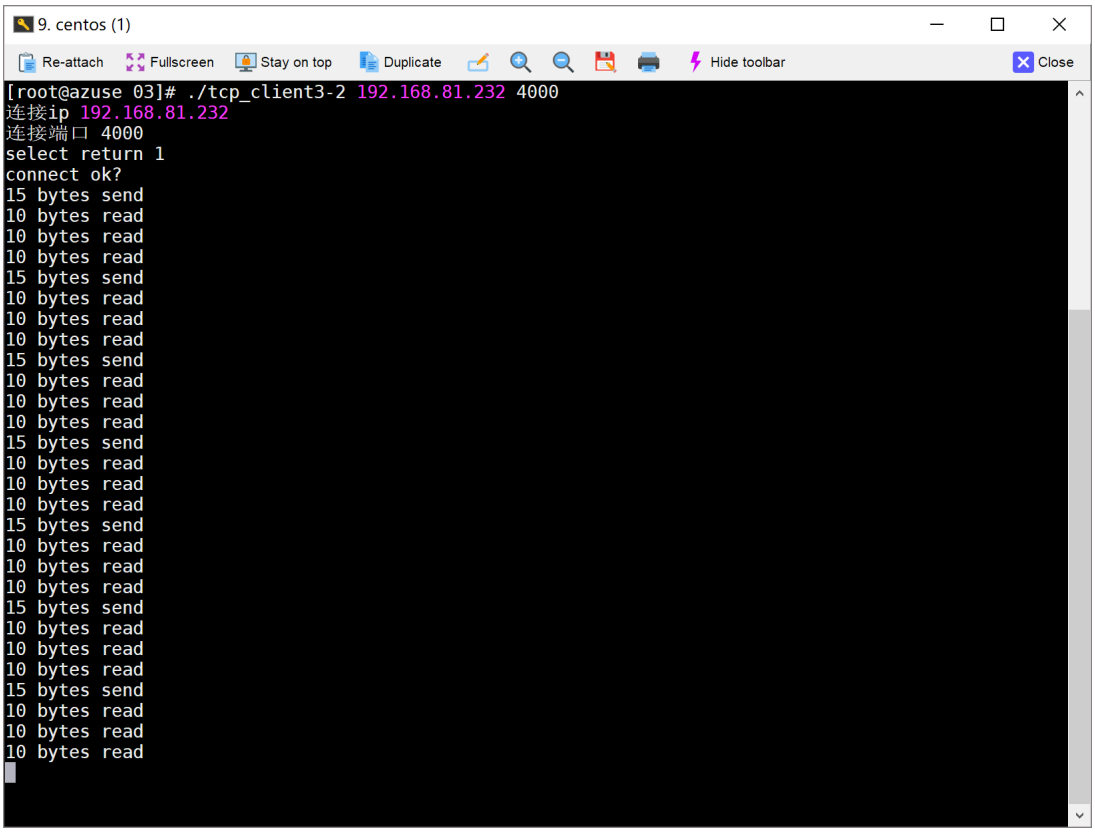
```
//server
...
        opt = 88;
        if (setsockopt(new_socket, SOL_SOCKET, SO_RCVLOWAT, &opt, sizeof(opt)))
        {
                perror("setsockopt");
                exit(EXIT_FAILURE);
        }
        int recv_min_size1;
        socklen_t recv_min_len = sizeof(recv_min_size1);
        int flag = getsockopt(new_socket, SOL_SOCKET, SO_RCVLOWAT, (void
*)&recv_min_size1, &recv_min_len);
        if (flag >= 0)
                printf("set SO_RCVLOWAT OK!!! SO_RCVLOWAT = %d\n",
recv_min_size1);
        else
                printf("set SO_RCVLOWAT failed!!! SO_RCVLOWAT = %d\n",
recv_min_size1);
...

...
        selres = select(new_socket + 1, &rfds2, NULL, NULL, NULL);
        if (selres <= 0)
        {
                // printf("select return %d\n", selres);
                continue;
        }
        else
        {
                printf("select return %d\n", selres);
        }
        int read = recv(new_socket, buffer, 88, 0);
        if (read == 0)
        {
                perror("Connect closed");
                break;
        }
        else if (read < 0)
        {
                continue;
                printf("recv return : %d\n", read);
                if (errno == EPIPE || errno == EWOULDBLOCK || errno == EAGAIN)
                {
                        perror("");
                        continue;
                }
                else
                {
                        perror("");
                        break;
                }
```

```
        }
        else
        {
                printf("%d bytes read\n", read);
        }
```

server每次读取88字节：



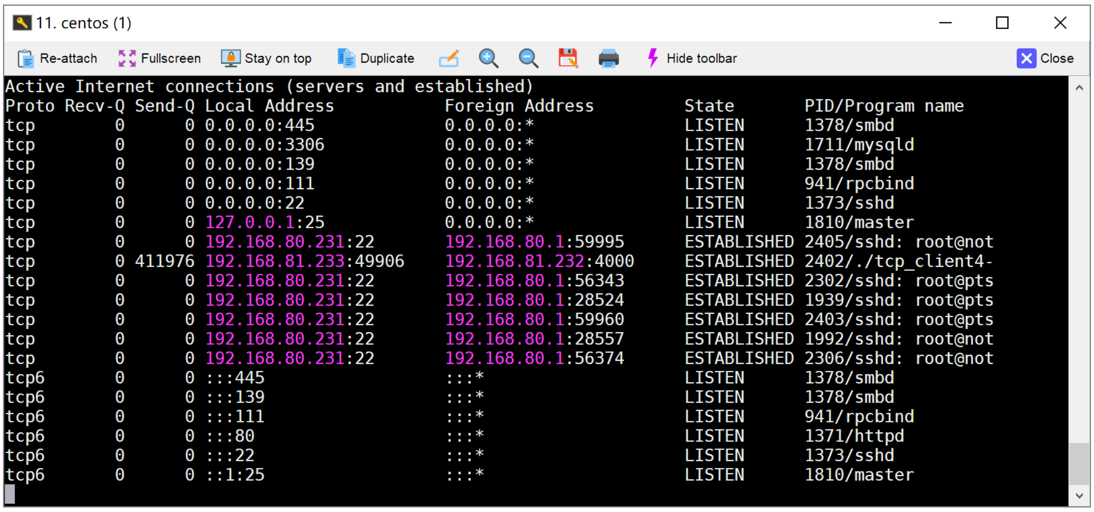
client:



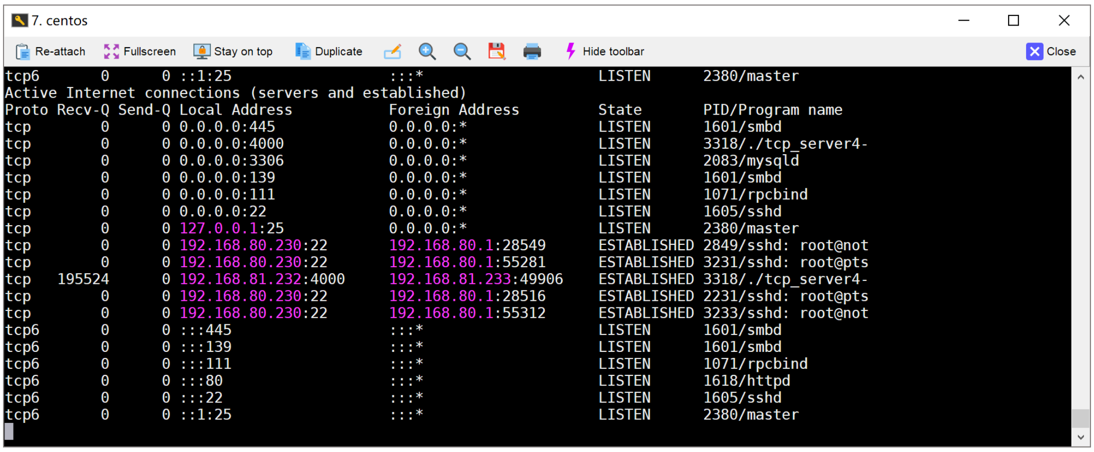
04

tcp4-1 server在连接成功后暂停，client死循环write，观察netstat

client一段时间后send返回-1，client的sendq与server的recvq满

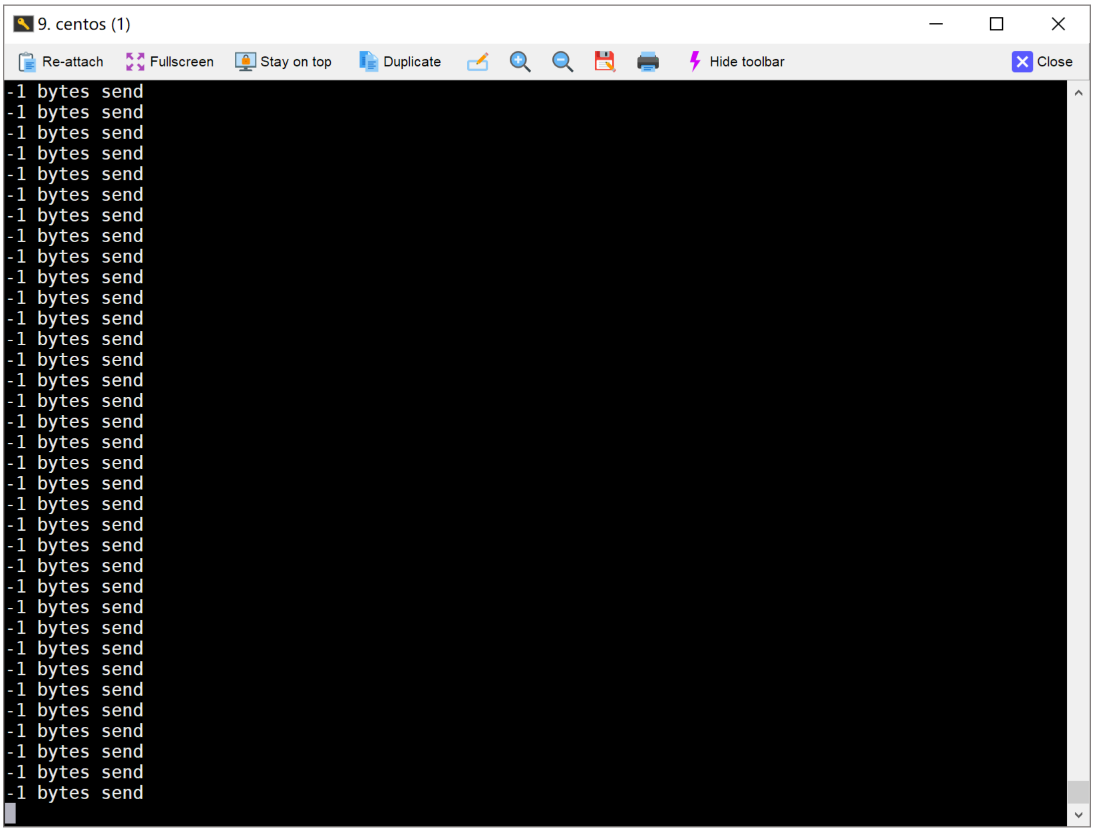


```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      1378/smbd
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      1711/mysqld
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      1378/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      941/rpcbind
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1373/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1810/master
tcp        0      0 192.168.80.231:22       192.168.80.1:59995      ESTABLISHED 2405/sshd: root@not
tcp        0 411976 192.168.81.233:49906    192.168.81.232:4000     ESTABLISHED 2402/./tcp_client4-
tcp        0      0 192.168.80.231:22       192.168.80.1:56343      ESTABLISHED 2302/sshd: root@pts
tcp        0      0 192.168.80.231:22       192.168.80.1:28524      ESTABLISHED 1939/sshd: root@pts
tcp        0      0 192.168.80.231:22       192.168.80.1:59960      ESTABLISHED 2403/sshd: root@pts
tcp        0      0 192.168.80.231:22       192.168.80.1:28557      ESTABLISHED 1992/sshd: root@not
tcp        0      0 192.168.80.231:22       192.168.80.1:56374      ESTABLISHED 2306/sshd: root@not
tcp6       0      0 :::445                  :::*                    LISTEN      1378/smbd
tcp6       0      0 :::139                  :::*                    LISTEN      1378/smbd
tcp6       0      0 :::111                  :::*                    LISTEN      941/rpcbind
tcp6       0      0 :::80                   :::*                    LISTEN      1371/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      1373/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1810/master
```



```
tcp6       0      0 ::1:25                  :::*                    LISTEN      2380/master
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      1601/smbd
tcp        0      0 0.0.0.0:4000            0.0.0.0:*               LISTEN      3318/./tcp_server4-
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      2083/mysqld
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      1601/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      1071/rpcbind
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1605/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      2380/master
tcp        0      0 192.168.80.230:22       192.168.80.1:28549      ESTABLISHED 2849/sshd: root@not
tcp        0      0 192.168.80.230:22       192.168.80.1:55281      ESTABLISHED 3231/sshd: root@pts
tcp   195524      0 192.168.81.232:4000     192.168.81.233:49906    ESTABLISHED 3318/./tcp_server4-
tcp        0      0 192.168.80.230:22       192.168.80.1:28516      ESTABLISHED 2231/sshd: root@pts
tcp        0      0 192.168.80.230:22       192.168.80.1:55312      ESTABLISHED 3233/sshd: root@not
tcp6       0      0 :::445                  :::*                    LISTEN      1601/smbd
tcp6       0      0 :::139                  :::*                    LISTEN      1601/smbd
tcp6       0      0 :::111                  :::*                    LISTEN      1071/rpcbind
tcp6       0      0 :::80                   :::*                    LISTEN      1618/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      1605/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      2380/master
```

client端send返回-1:

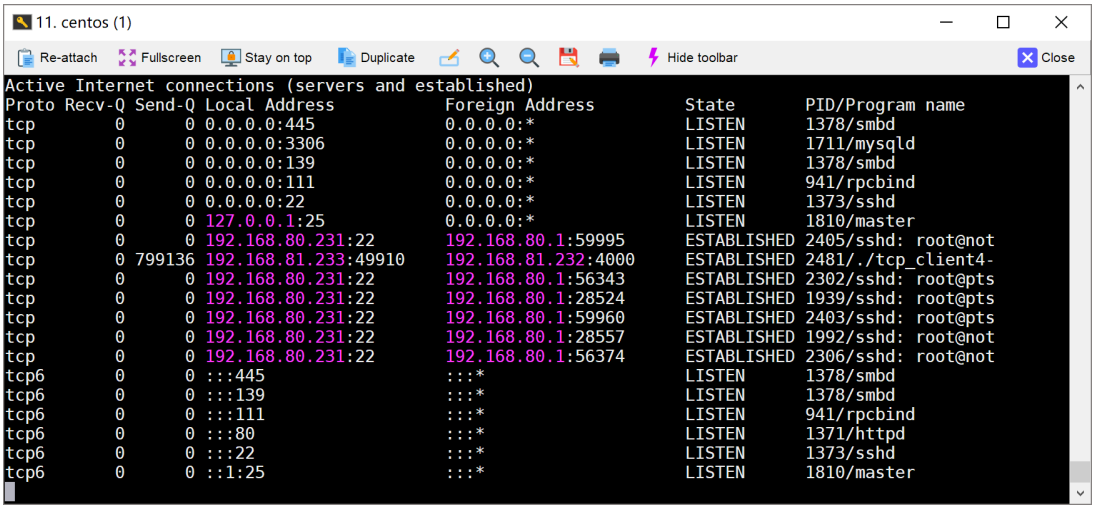


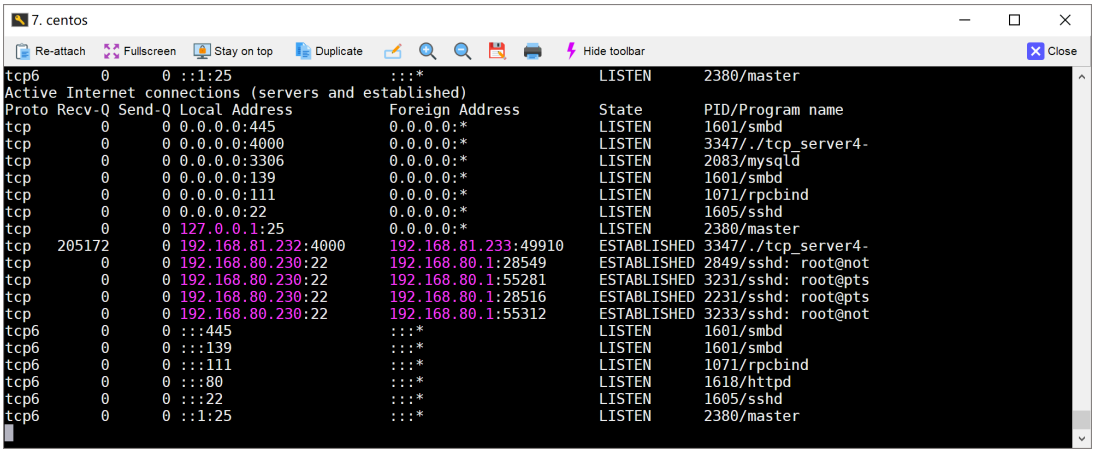
```
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
-1 bytes send
```

server端暂停:

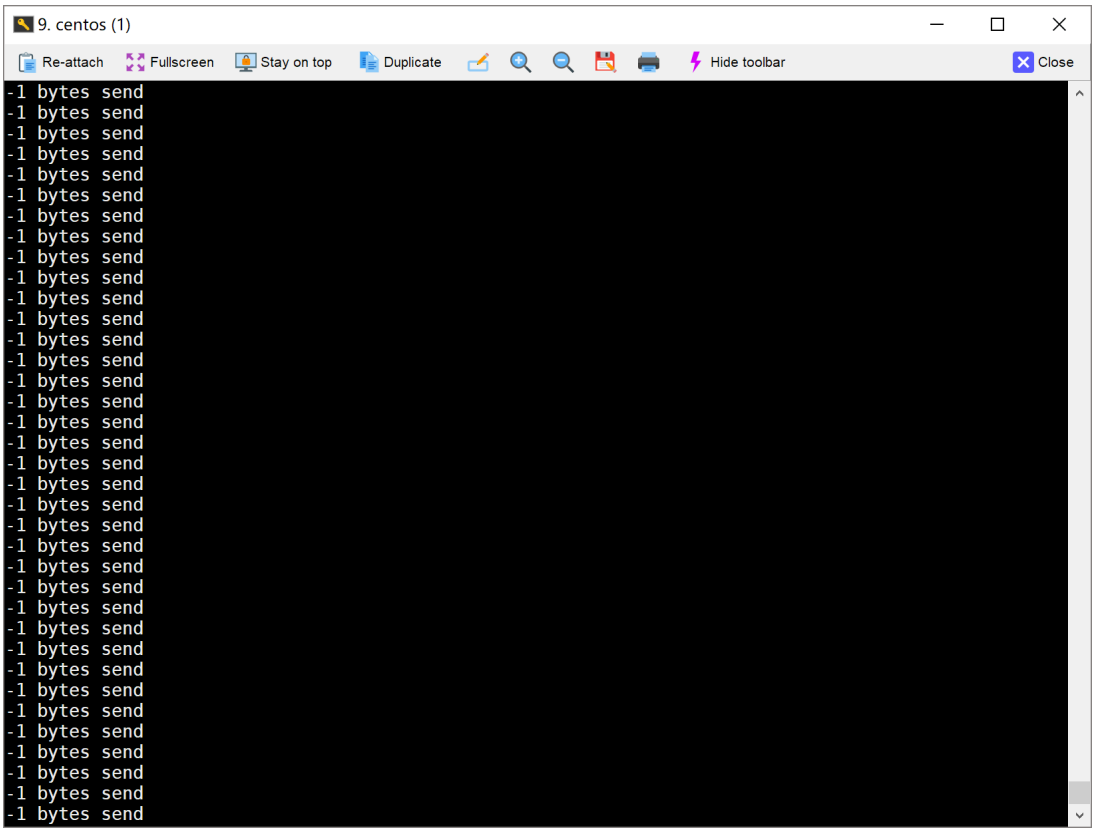
tcp4-2 server在连接成功后每隔1s读20字节，client死循环写，观察netstat，write失败后，如何重新恢复为继续写？

netstat结果：client的sendq满，server的recvq满





client写失败：



server每隔1s读取20字节:

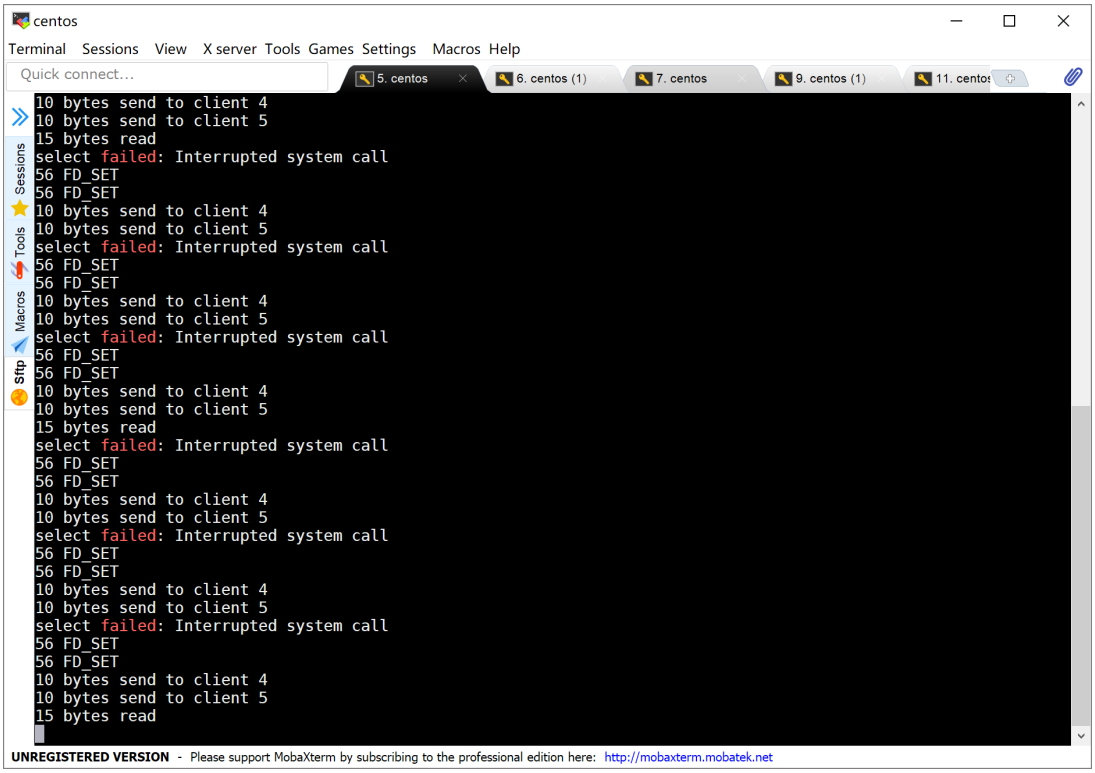
如何重新恢复为继续写?

在write失败后`select(socket, NULL, &wfds, NULL, NULL);`等待select返回>0并且判断端口合法后可写。

05 一个非阻塞server连接多个client

在server程序中将server监听socket和所有client连接socket都放在rfds中，程序主体为一个循环，循环内先看有没有要accept的新连接，没有的话就read。write当每次SIGALRM打断时设置write_flag，在循环主体中write。

server端接收多个client（两个client编号分别为4和5）：



server端的主循环：

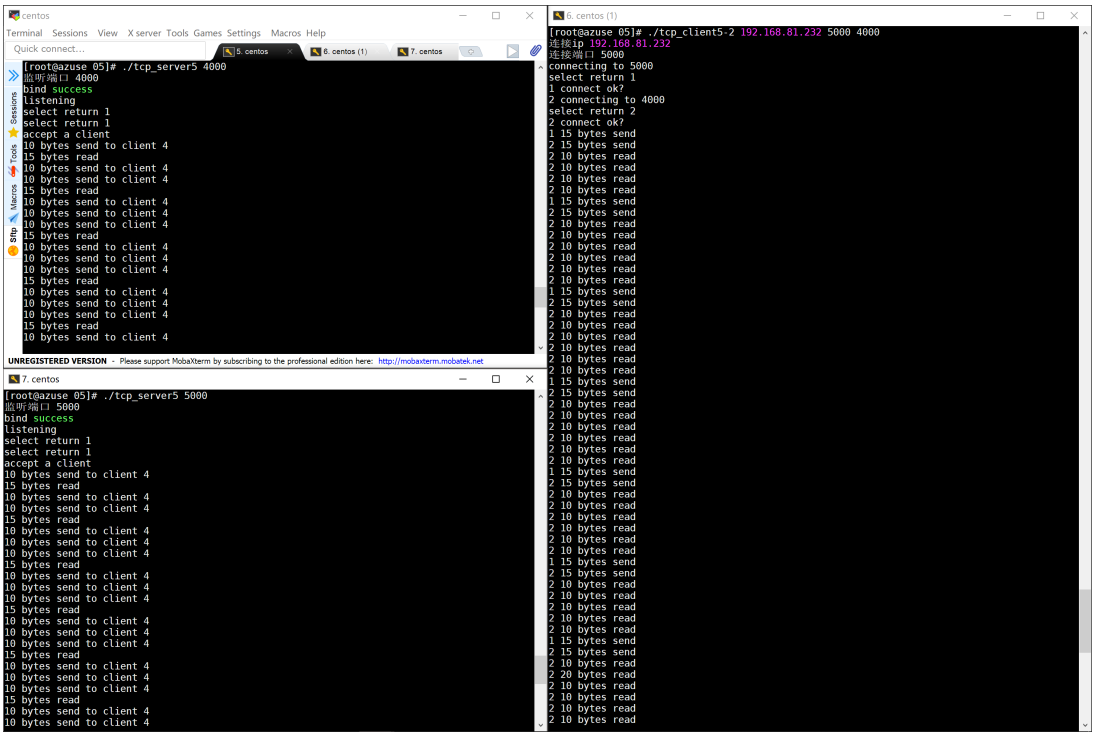
```
while (1)
                {
                        //检查每个socket，并进行读(如果是sock则accept)
                        int i;
                        for (i = 0; i <= maxfd; i++)
                        {
                                //printf("%d\n",i);
                                if (!FD_ISSET(i, &rfds))
                                {
                                        //printf("%d rfds not set\n",i);
                                        continue;
                                }
                                //可读的socket

                                if(i > server_fd)
                                {
                                        //当前是client连接的socket，可以写(read
from client)
                                        if (write_flag)
                                        {
                                                write_flag = 0;
                                                int j;
                                                for (j = 0; j <= maxfd; j++)
                                                {
```

```
                                                        if(j <= server_fd)
                                                                continue;

                                                        int send_bytes = send(j,
buffer_send, 10, 0);
                                                        printf("%d bytes send to
client %d\n", send_bytes, j);
                                                }
                                                alarm(1);
                                        }

                                        int read = recv(new_socket, buffer, 100,
0);
                                        if (read == 0)
                                        {
                                                perror("Connect closed");
                                                break;
                                        }
                                        else if (read < 0)
                                        {
                                                continue;
                                                printf("recv return : %d\n",
read);
                                                if (errno == EPIPE || errno ==
EWOULDBLOCK || errno == EAGAIN)
                                                {
                                                        perror("");
                                                        continue;
                                                }
                                                else
                                                {
                                                        perror("");
                                                        break;
                                                }
                                        }
                                        else
                                        {
                                                printf("%d bytes read\n", read);
                                        }
                                }
                                else if (i == server_fd)
                                {
                                        tv.tv_sec = 1;
                                        tv.tv_usec = 0;
                                        selres = select(server_fd + 1, &rfds,
NULL, NULL, &tv);
                                        if (selres == -1)
                                        {
                                                perror("select failed");
                                                int k;
                                                for(k = server_fd + 1; k <= maxfd;
k++)
                                                {
                                                        FD_SET(k, &rfds);
```

```
                                                        printf("%d FD_SET\n");
                                                }
                                                continue;
                                        }
                                        else if (selres == 0)
                                        {
                                                continue;
                                        }

                                        //当前是server的socket，不进行读写而是accept
新连接
                                        new_socket = accept(server_fd, (struct
sockaddr *)&address, (socklen_t *)&addrlen);
                                        if (new_socket == -1)
                                        {
                                                perror("accept failed");
                                                continue;
                                        }

                                        printf("accept another client\n");

                                        //设置new_sock为non-blocking
                                        flags = fcntl(new_socket, F_GETFL, 0);
                                        fcntl(new_socket, F_SETFL, flags |
O_NONBLOCK);

                                        //把new_sock添加到select的侦听中
                                        if (new_socket > maxfd)
                                        {
                                                maxfd = new_socket;
                                        }

                                        FD_SET(new_socket, &rfds);
                                }
                        }
                }
```

tcp5-2 启动两个server5 用一个client连接

在client中执行两遍非阻塞的connect过程分别连接两个端口即可



```
[root@azuse 05]# ./tcp_server5 4000
监听端口 4000
bind success
listening
select return 1
select return 1
accept a client
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

```
[root@azuse 05]# ./tcp_server5 5000
监听端口 5000
bind success
listening
select return 1
select return 1
accept a client
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
10 bytes send to client 4
15 bytes read
10 bytes send to client 4
10 bytes send to client 4
```

```
[root@azuse 05]# ./tcp_client5-2 192.168.81.232 5000 4000
连接ip 192.168.81.232
连接端口 5000
connecting to 5000
select return 1
1 connect ok?
2 connecting to 4000
select return 2
2 connect ok?
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
1 15 bytes send
2 15 bytes send
2 10 bytes read
2 20 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
2 10 bytes read
```