# 计算机网络实验报告

| | |
|---|---|
| **实验：** | 实验 3 Socket 编程 |
| **专业：** | 计算机科学与技术 |
| **班级：** | 1 班 |
| **姓名：** | 姚怀聿 |
| **学号：** | 22920202204632 |

2022 年 11 月 2 日

# 目　　　录

## 一、实验目的

1. 掌握 TCP 和 UDP 协议主要特点

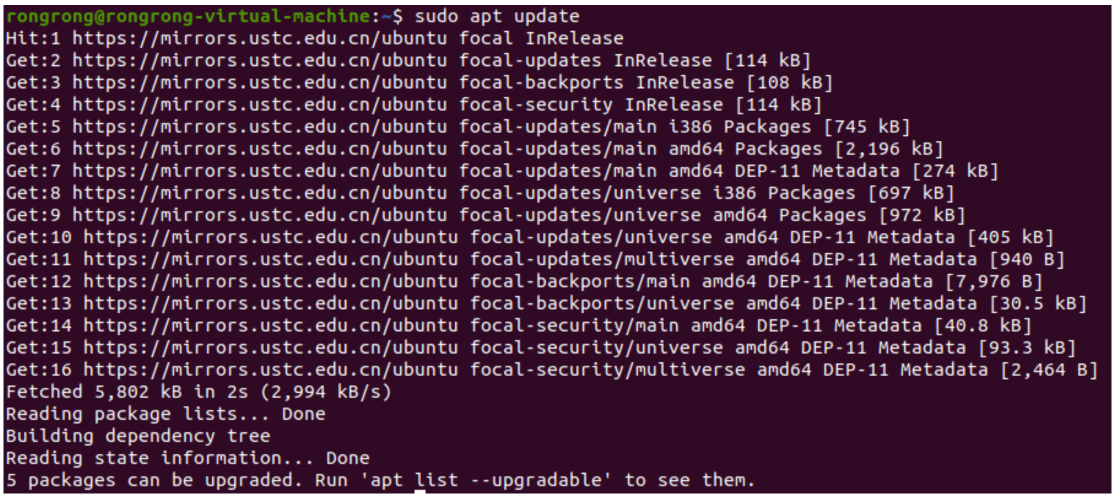
2. 理解 socket 的基本概念和工作原理，编程实现 socket 通信

## 二、实验内容

### 0. 准备工作

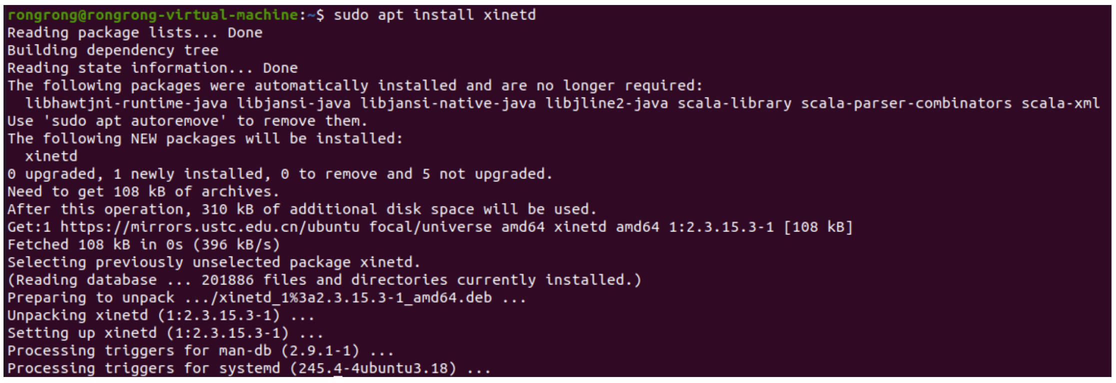
1. 使用终端命令行安装本地 echo 服务，监听 TCP 7 号端口

1) `sudo apt update`



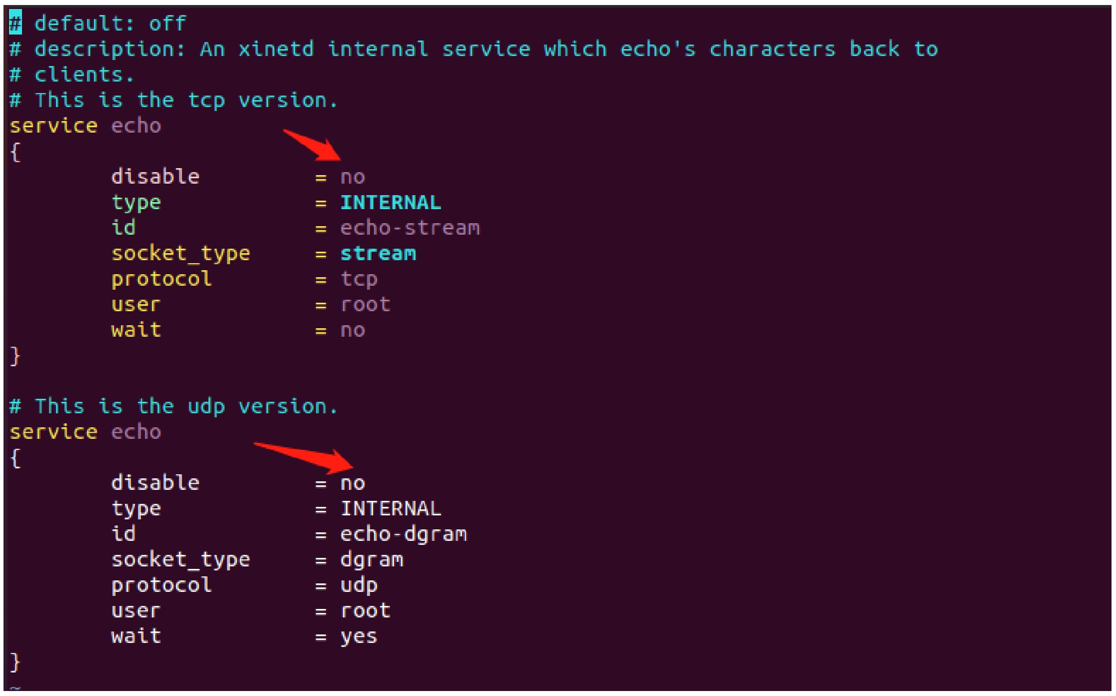
```
rongrong@rongrong-virtual-machine:~$ sudo apt update
Hit:1 https://mirrors.ustc.edu.cn/ubuntu focal InRelease
Get:2 https://mirrors.ustc.edu.cn/ubuntu focal-updates InRelease [114 kB]
Get:3 https://mirrors.ustc.edu.cn/ubuntu focal-backports InRelease [108 kB]
Get:4 https://mirrors.ustc.edu.cn/ubuntu focal-security InRelease [114 kB]
Get:5 https://mirrors.ustc.edu.cn/ubuntu focal-updates/main i386 Packages [745 kB]
Get:6 https://mirrors.ustc.edu.cn/ubuntu focal-updates/main amd64 Packages [2,196 kB]
Get:7 https://mirrors.ustc.edu.cn/ubuntu focal-updates/main amd64 DEP-11 Metadata [274 kB]
Get:8 https://mirrors.ustc.edu.cn/ubuntu focal-updates/universe i386 Packages [697 kB]
Get:9 https://mirrors.ustc.edu.cn/ubuntu focal-updates/universe amd64 Packages [972 kB]
Get:10 https://mirrors.ustc.edu.cn/ubuntu focal-updates/universe amd64 DEP-11 Metadata [405 kB]
Get:11 https://mirrors.ustc.edu.cn/ubuntu focal-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:12 https://mirrors.ustc.edu.cn/ubuntu focal-backports/main amd64 DEP-11 Metadata [7,976 B]
Get:13 https://mirrors.ustc.edu.cn/ubuntu focal-backports/universe amd64 DEP-11 Metadata [30.5 kB]
Get:14 https://mirrors.ustc.edu.cn/ubuntu focal-security/main amd64 DEP-11 Metadata [40.8 kB]
Get:15 https://mirrors.ustc.edu.cn/ubuntu focal-security/universe amd64 DEP-11 Metadata [93.3 kB]
Get:16 https://mirrors.ustc.edu.cn/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Fetched 5,802 kB in 2s (2,994 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

2) `sudo apt install xinetd`



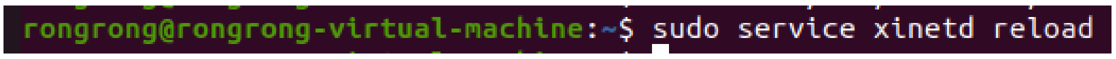
```
rongrong@rongrong-virtual-machine:~$ sudo apt install xinetd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libhawtjni-runtime-java libjansi-java libjansi-native-java libjline2-java scala-library scala-parser-combinators scala-xml
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  xinetd
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 108 kB of archives.
After this operation, 310 kB of additional disk space will be used.
Get:1 https://mirrors.ustc.edu.cn/ubuntu focal/universe amd64 xinetd amd64 1:2.3.15.3-1 [108 kB]
Fetched 108 kB in 0s (396 kB/s)
Selecting previously unselected package xinetd.
(Reading database ... 201886 files and directories currently installed.)
Preparing to unpack .../xinetd_1%3a2.3.15.3-1_amd64.deb ...
Unpacking xinetd (1:2.3.15.3-1) ...
Setting up xinetd (1:2.3.15.3-1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.18) ...
```

3）修改/etc/xinetd.d/echo 文件中的内容，将 disable 设置为 no



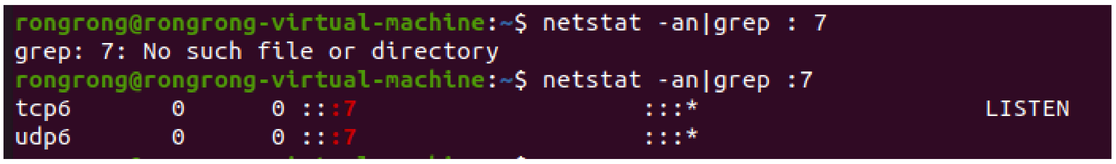
```
# default: off
# description: An xinetd internal service which echo's characters back to
# clients.
# This is the tcp version.
service echo
{
        disable         = no
        type            = INTERNAL
        id              = echo-stream
        socket_type     = stream
        protocol        = tcp
        user            = root
        wait            = no
}

# This is the udp version.
service echo
{
        disable         = no
        type            = INTERNAL
        id              = echo-dgram
        socket_type     = dgram
        protocol        = udp
        user            = root
        wait            = yes
}
```

4）`sudo service xinetd reload`



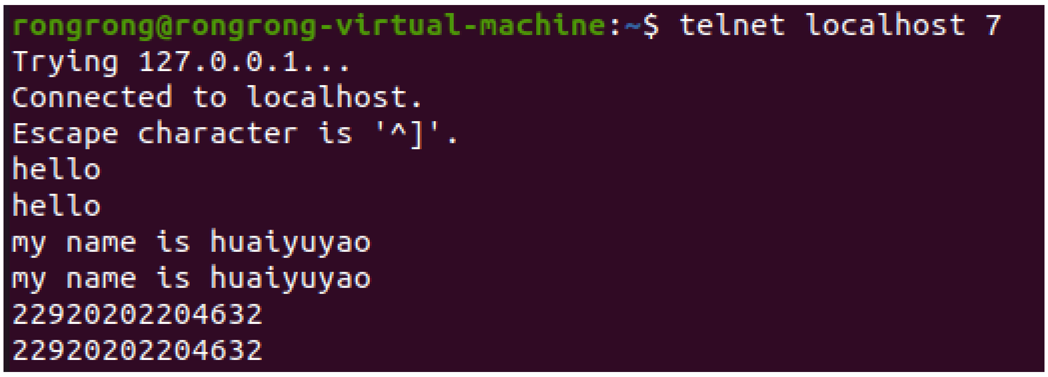
```
rongrong@rongrong-virtual-machine:~$ sudo service xinetd reload
```

5）`netstat -an|grep :7`



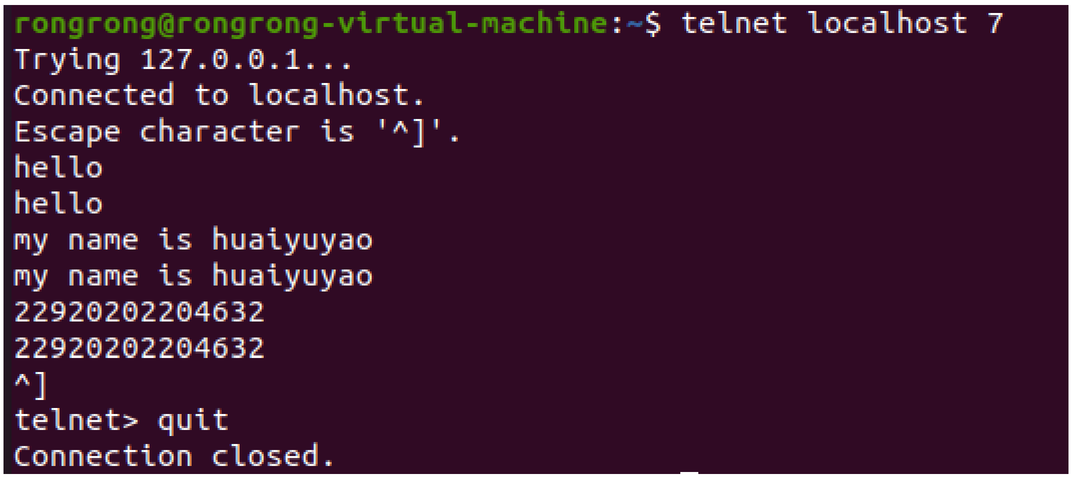
```
rongrong@rongrong-virtual-machine:~$ netstat -an|grep : 7
grep: 7: No such file or directory
rongrong@rongrong-virtual-machine:~$ netstat -an|grep :7
tcp6       0      0 :::7                    :::*                    LISTEN
udp6       0      0 :::7                    :::*
```

2. 输入`telnet localhost 7`，连接本机 echo 服务器，输入任意文本，观察响应



```
rongrong@rongrong-virtual-machine:~$ telnet localhost 7
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
hello
hello
my name is huaiyuyao
my name is huaiyuyao
22920202204632
22920202204632
```

3. 输入`Ctrl+]`结束 echo 服务，输入`close`或`quit`退出 telnet

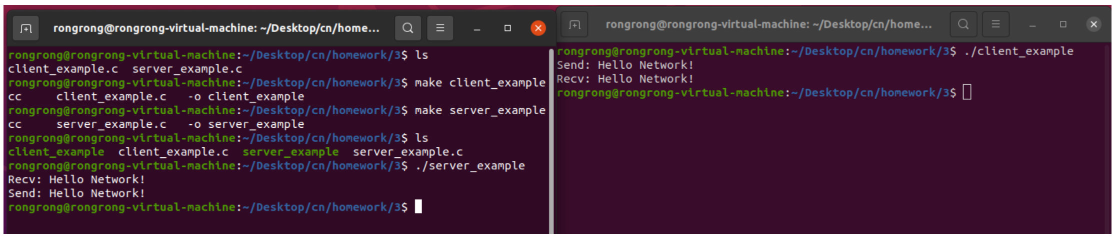


```
rongrong@rongrong-virtual-machine:~$ telnet localhost 7
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
hello
hello
my name is huaiyuyao
my name is huaiyuyao
22920202204632
22920202204632
^]
telnet> quit
Connection closed.
```

## 1. 完善 socket 客户机

1) 开启两个终端窗口，分别编译、运行 server_example.c 和 client_example.c，观察它们实现的功能

i. 编译、运行 server_example.c 和 client_example.c

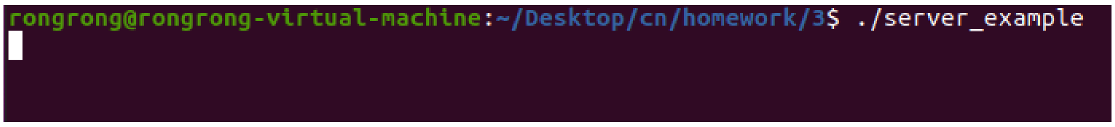


```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ls
client_example.c  server_example.c
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ make client_example
cc     client_example.c   -o client_example
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ make server_example
cc     server_example.c   -o server_example
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ls
client_example  client_example.c  server_example  server_example.c
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
Recv: Hello Network!
Send: Hello Network!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$
```

```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client_example
Send: Hello Network!
Recv: Hello Network!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$
```

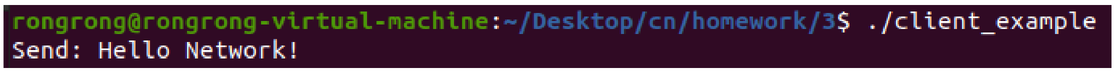
ii. 观察它们实现的功能

首先运行 server_example，终端不会显示任何内容，处于 server 端处于监听状态



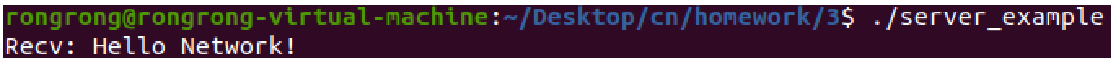
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
```

然后再运行 client_example，client 端会向 server 端发送一个字符串 “Hello Network!”，并在终端上显示出该条发送的信息



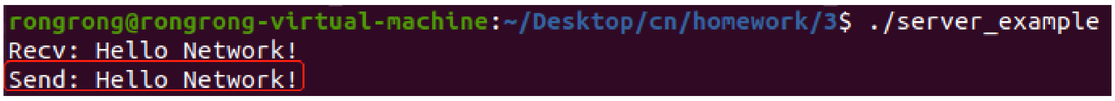
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client_example
Send: Hello Network!
```

server 端接到该条信息后在终端上显示接收到这条信息



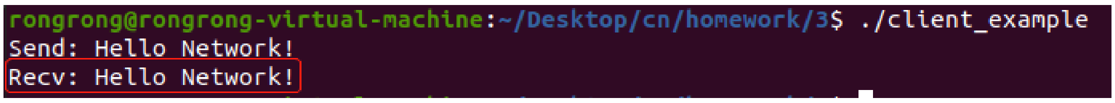
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
Recv: Hello Network!
```

server 端同时向 client 端发送一条字符串 “Hello Network!”，并在终端上显示发送了这条字符串的信息



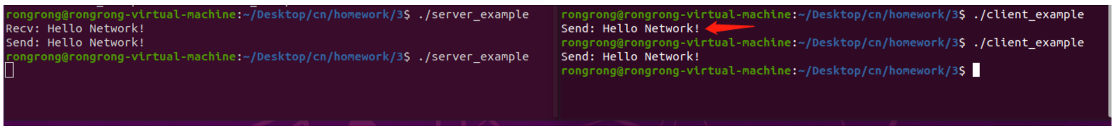
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
Recv: Hello Network!
Send: Hello Network!
```

client 端收到 server 端发送过来的字符串，在终端上输出接收到该条字符串的信息



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client_example
Send: Hello Network!
Recv: Hello Network!
```

倘若将运行这两个程序的顺序颠倒一下，就会产生以下效果：



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
Recv: Hello Network!
Send: Hello Network!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server_example
```

```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client_example
Send: Hello Network!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client_example
Send: Hello Network!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$
```

可以看到，client 端发送一个字符串后无法收到来自 server 端的回

复；而如果之后再打开 server 端，也无法将二者建立起连接，无论 client 端再发送多少次，server 端也无法接收到来自 client 端的信息。

2) 按以下要求，修改范例 client_example.c，实现类似 telnet 连接 echo 服务器的效果

i. 为所有 socket 函数调用添加错误处理代码

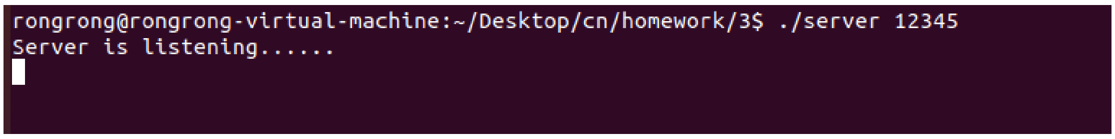
ii. 范例中服务器地址和端口是固定值，请将它们改成允许用户以命令行参数形式输入

iii. 范例中客户机发送的是固定文本“Hello Network!”，请改成允许用户输入字符串，按回车发送

iv. 实现循环，直至客户机输入“bye”退出

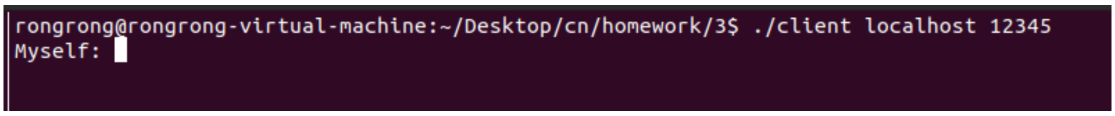
实验结果：

终端使用命令`./server 12345`指定端口号 12345 运行服务端：



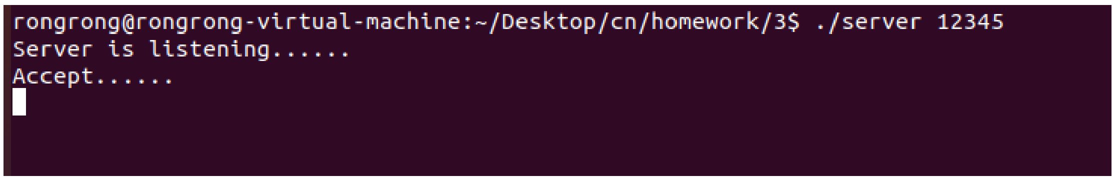
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
```

使用命令`./client localhost 12345`指定地址和端口号运行客户端：



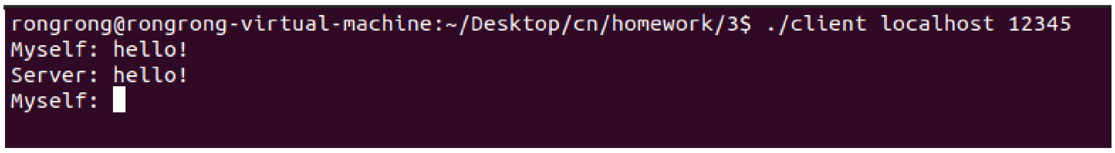
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 12345
Myself: 
```

可以看到，server 端变为接收状态：



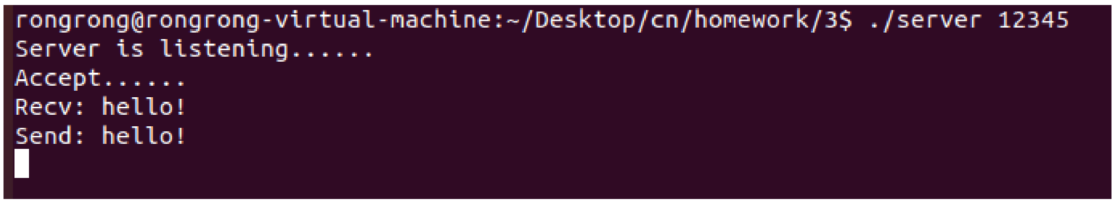
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
Accept......
```

向客户机输入字符串：



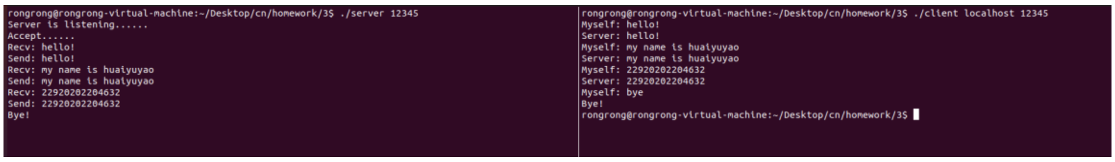
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 12345
Myself: hello!
Server: hello!
Myself: 
```

几乎同一时刻，可以看到 server 端接收到信息并向终端输出信息



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
Accept......
Recv: hello!
Send: hello!
```

在 client 端输入“bye”后，进程终止，并向终端输出“Bye!”以示结束：



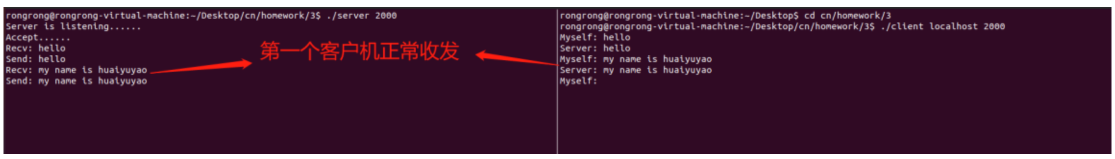
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
Accept......
Recv: hello!
Send: hello!
Recv: my name is huaiyuyao
Send: my name is huaiyuyao
Recv: 22920202204632
Send: 22920202204632
Bye!
```

```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 12345
Myself: hello!
Server: hello!
Myself: my name is huaiyuyao
Server: my name is huaiyuyao
Myself: 22920202204632
Server: 22920202204632
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ 
```

## 2. 字符串回送服务器（TCP 迭代）

1） 客户机和服务器的运行情况

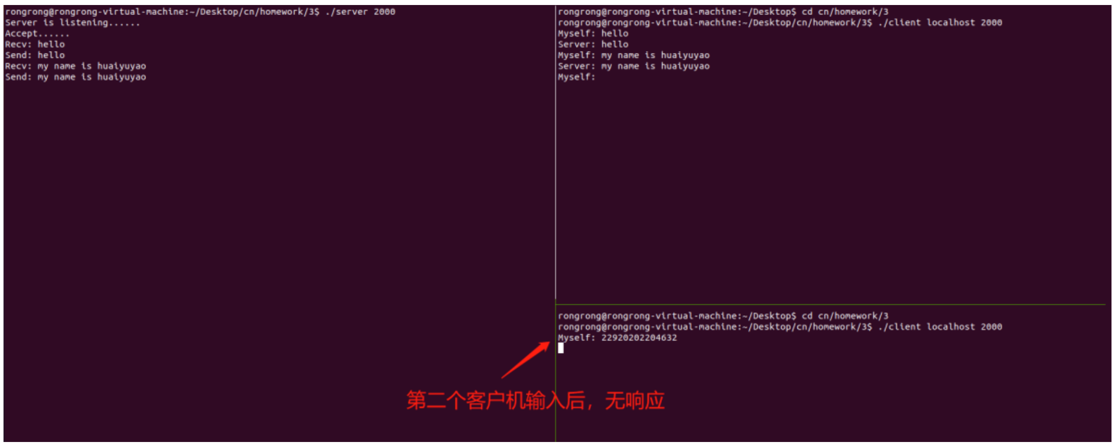
第一个客户机正常收发：



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 2000
Server is listening......
Accept......
Recv: hello
Send: hello
Recv: my name is huaiyuyao
Send: my name is huaiyuyao
```

```
rongrong@rongrong-virtual-machine:~/Desktop$ cd cn/homework/3
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 2000
Myself: hello
Server: hello
Myself: my name is huaiyuyao
Server: my name is huaiyuyao
Myself:
```

第 2 个客户机输入后，无响应：

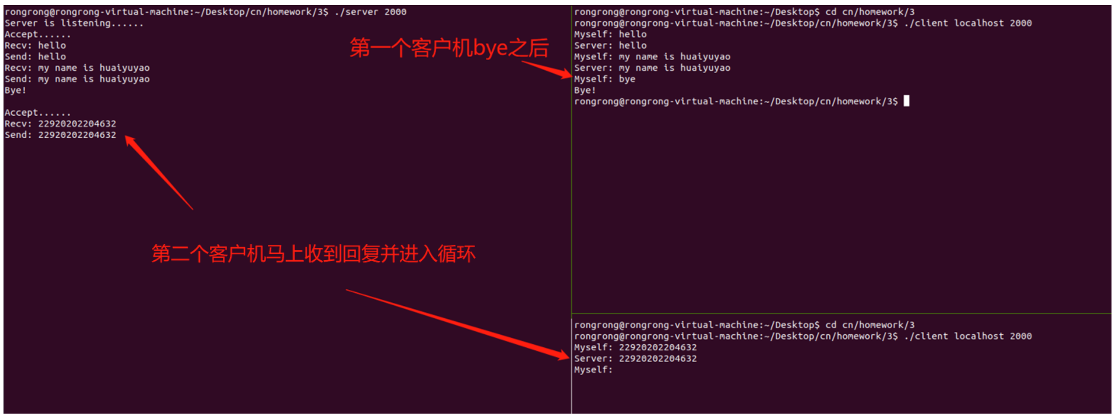


```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 2000
Server is listening......
Accept......
Recv: hello
Send: hello
Recv: my name is huaiyuyao
Send: my name is huaiyuyao
```

```
rongrong@rongrong-virtual-machine:~/Desktop$ cd cn/homework/3
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 2000
Myself: hello
Server: hello
Myself: my name is huaiyuyao
Server: my name is huaiyuyao
Myself:
```

```
rongrong@rongrong-virtual-machine:~/Desktop$ cd cn/homework/3
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 2000
Myself: 22920202204632
```

第 1 个客户机 bye 之后，第 2 个客户机马上收到回复并进入循环：

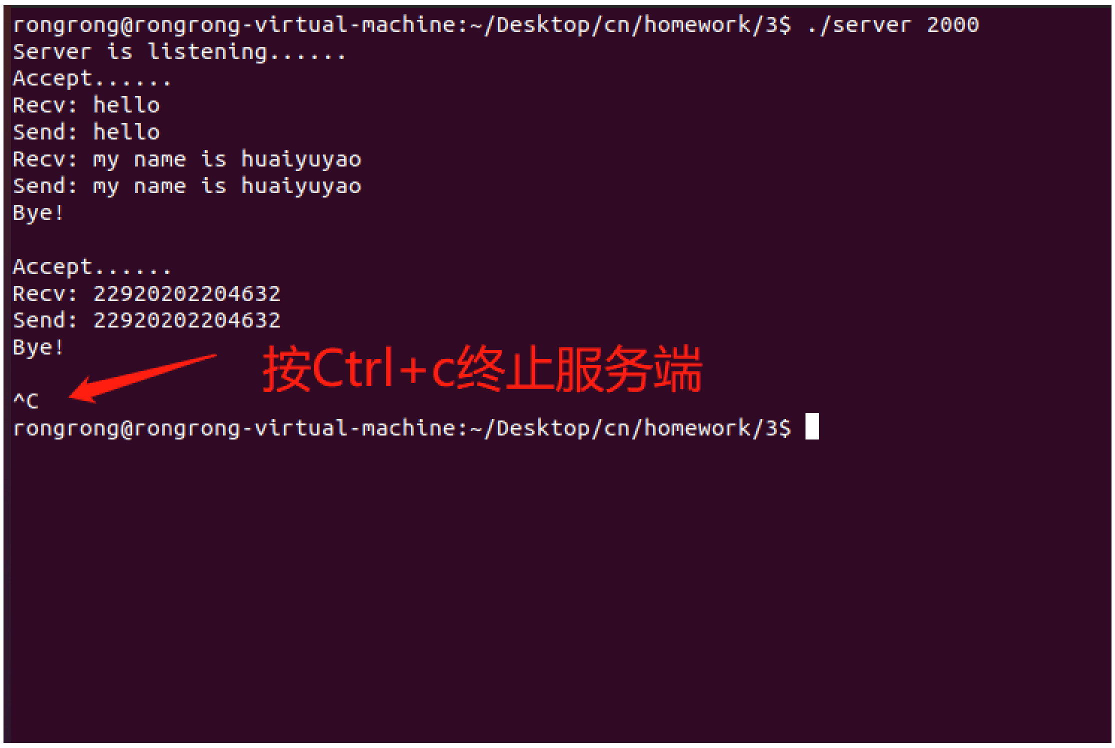


```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 2000
Server is listening......
Accept......
Recv: hello
Send: hello
Recv: my name is huaiyuyao
Send: my name is huaiyuyao
Bye!

Accept......
Recv: 22920202204632
Send: 22920202204632
```

```
rongrong@rongrong-virtual-machine:~/Desktop$ cd cn/homework/3
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 2000
Myself: hello
Server: hello
Myself: my name is huaiyuyao
Server: my name is huaiyuyao
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$
```

```
rongrong@rongrong-virtual-machine:~/Desktop$ cd cn/homework/3
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./client localhost 2000
Myself: 22920202204632
Server: 22920202204632
Myself:
```

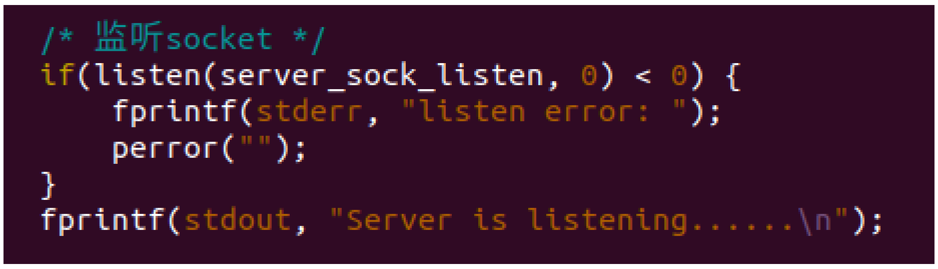
按 Ctrl+c 终止服务器程序：



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 2000
Server is listening......
Accept......
Recv: hello
Send: hello
Recv: my name is huaiyuyao
Send: my name is huaiyuyao
Bye!

Accept......
Recv: 22920202204632
Send: 22920202204632
Bye!
^C
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$
```

2) 设置服务器 listen( )的 backlog 为 0 或 1，同时打开多个终端窗口、让 4 个客户机连接服务器，使用 netstat 命令观察 socket 状态变化，对照 TCP 连接状态图(5-28/5-29/5-30)，说明变化过程以及 backlog 与客户机完成队列的数量关系

I. backlog 设置为 0：

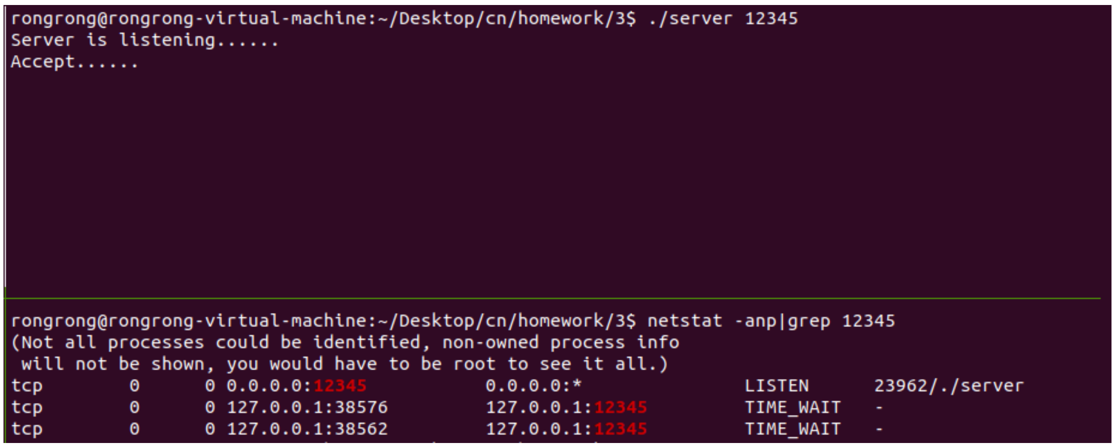


```
/* 监听socket */
if(listen(server_sock_listen, 0) < 0) {
    fprintf(stderr, "listen error: ");
    perror("");
}
fprintf(stdout, "Server is listening......\n");
```

让 4 个客户机连接服务器，使用 netstat 命令观察 socket 状态变化：

`netstat -anp | grep 12345`

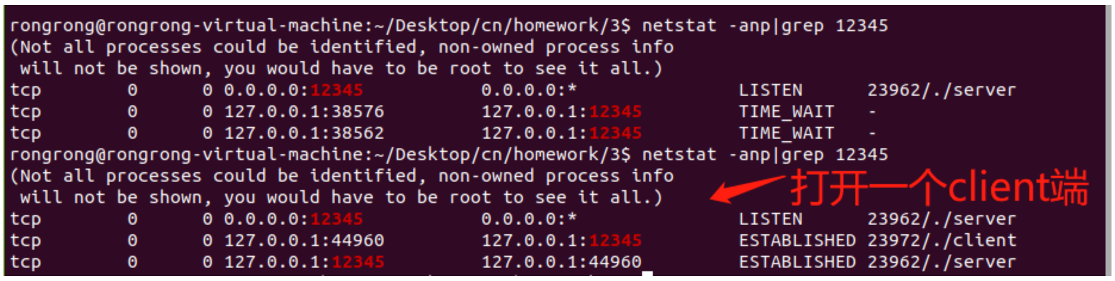
i.　只打开 server 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
Accept......
```

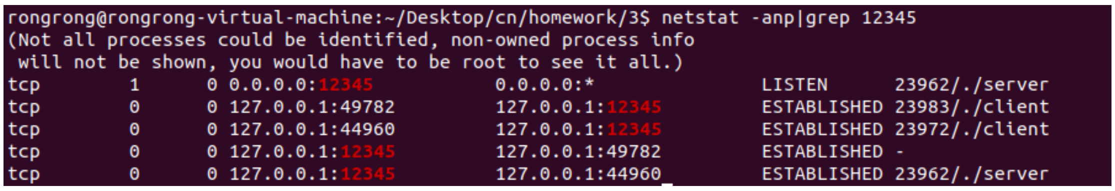
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:38576         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:38562         127.0.0.1:12345         TIME_WAIT   -
```

ii.　打开 1 个 client 端



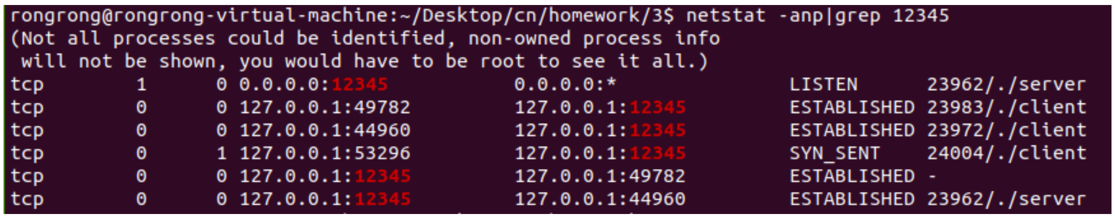
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:38576         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:38562         127.0.0.1:12345         TIME_WAIT   -
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:44960         127.0.0.1:12345         ESTABLISHED 23972/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:44960         ESTABLISHED 23962/./server
```

iii.　打开 2 个 client 端



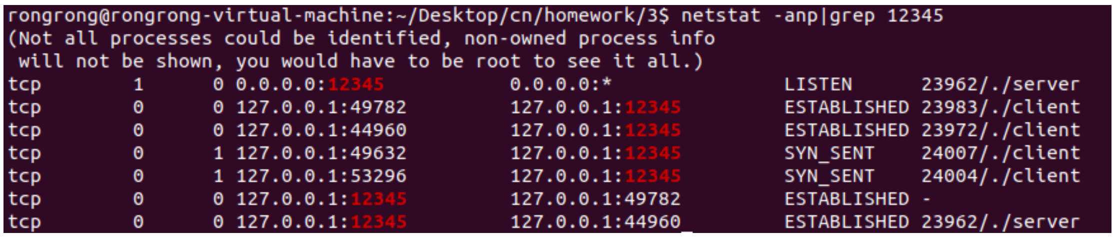
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        1      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:49782         127.0.0.1:12345         ESTABLISHED 23983/./client
tcp        0      0 127.0.0.1:44960         127.0.0.1:12345         ESTABLISHED 23972/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:49782         ESTABLISHED -
tcp        0      0 127.0.0.1:12345         127.0.0.1:44960         ESTABLISHED 23962/./server
```

iv.　打开 3 个 client 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        1      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:49782         127.0.0.1:12345         ESTABLISHED 23983/./client
tcp        0      0 127.0.0.1:44960         127.0.0.1:12345         ESTABLISHED 23972/./client
tcp        0      1 127.0.0.1:53296         127.0.0.1:12345         SYN_SENT    24004/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:49782         ESTABLISHED -
tcp        0      0 127.0.0.1:12345         127.0.0.1:44960         ESTABLISHED 23962/./server
```

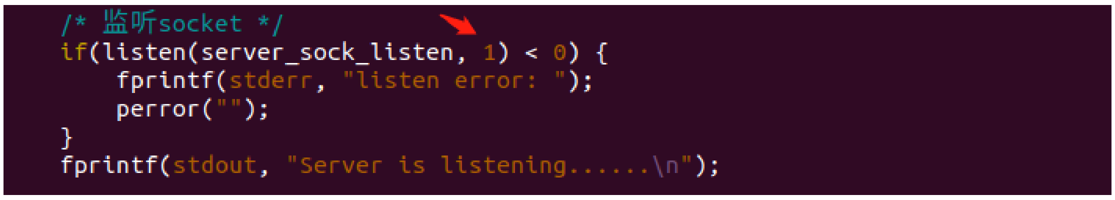
v.　打开 4 个 client 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        1      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      23962/./server
tcp        0      0 127.0.0.1:49782         127.0.0.1:12345         ESTABLISHED 23983/./client
tcp        0      0 127.0.0.1:44960         127.0.0.1:12345         ESTABLISHED 23972/./client
tcp        0      1 127.0.0.1:49632         127.0.0.1:12345         SYN_SENT    24007/./client
tcp        0      1 127.0.0.1:53296         127.0.0.1:12345         SYN_SENT    24004/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:49782         ESTABLISHED -
tcp        0      0 127.0.0.1:12345         127.0.0.1:44960         ESTABLISHED 23962/./server
```

同时打开 4 个客户端，有 2 个可以被建立并正常运行

II. backlog 设置为 1：

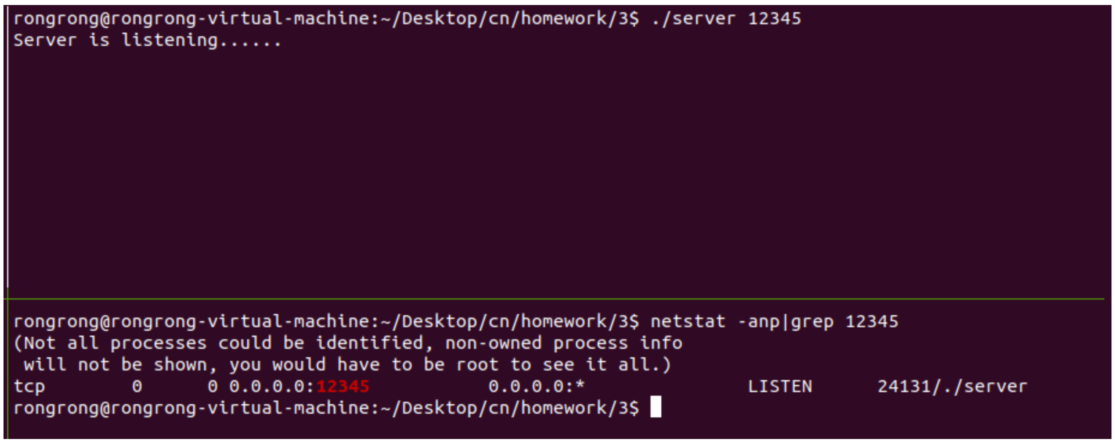


```
    /* 监听socket */
    if(listen(server_sock_listen, 1) < 0) {
        fprintf(stderr, "listen error: ");
        perror("");
    }
    fprintf(stdout, "Server is listening......\n");
```

让 4 个客户机连接服务器，使用 netstat 命令观察 socket 状态变化：

`netstat -anp | grep 12345`

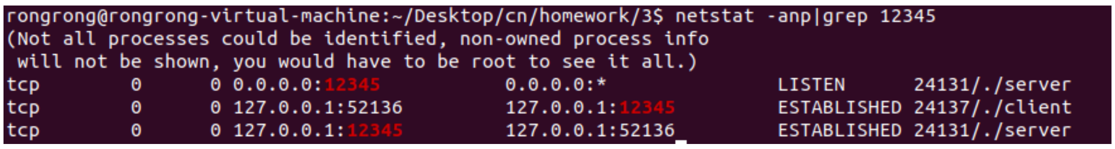
i. 只打开 server 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ ./server 12345
Server is listening......
```

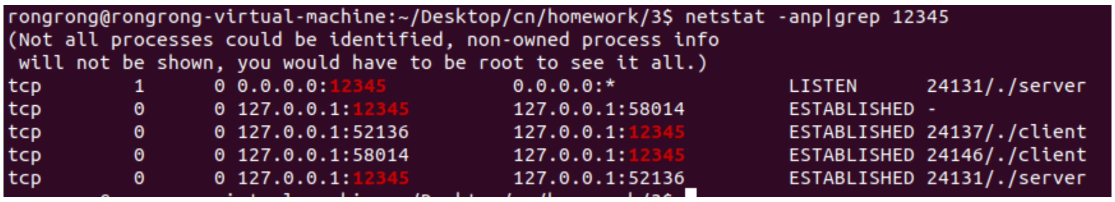
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      24131/./server
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ 
```

ii. 打开 1 个 client 端



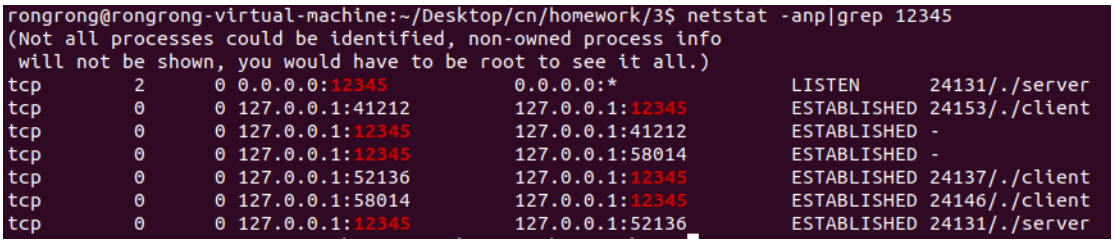
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      24131/./server
tcp        0      0 127.0.0.1:52136         127.0.0.1:12345         ESTABLISHED 24137/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:52136         ESTABLISHED 24131/./server
```

iii. 打开 2 个 client 端



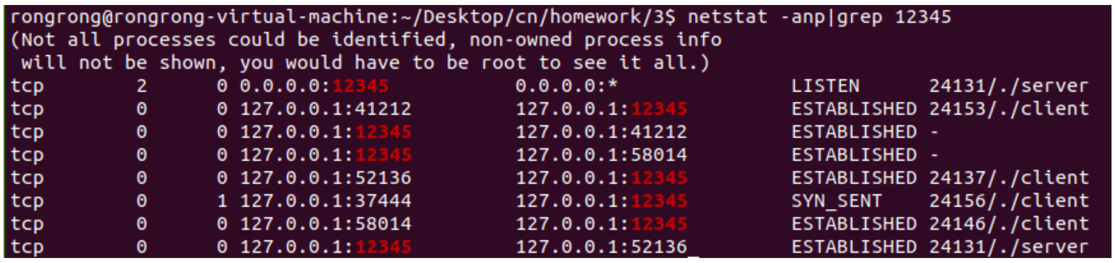
```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        1      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      24131/./server
tcp        0      0 127.0.0.1:12345         127.0.0.1:58014         ESTABLISHED -
tcp        0      0 127.0.0.1:52136         127.0.0.1:12345         ESTABLISHED 24137/./client
tcp        0      0 127.0.0.1:58014         127.0.0.1:12345         ESTABLISHED 24146/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:52136         ESTABLISHED 24131/./server
```

iv.　打开 3 个 client 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        2      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      24131/./server
tcp        0      0 127.0.0.1:41212         127.0.0.1:12345         ESTABLISHED 24153/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:41212         ESTABLISHED -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58014         ESTABLISHED -
tcp        0      0 127.0.0.1:52136         127.0.0.1:12345         ESTABLISHED 24137/./client
tcp        0      0 127.0.0.1:58014         127.0.0.1:12345         ESTABLISHED 24146/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:52136         ESTABLISHED 24131/./server
```

v.　打开 4 个 client 端



```
rongrong@rongrong-virtual-machine:~/Desktop/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        2      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      24131/./server
tcp        0      0 127.0.0.1:41212         127.0.0.1:12345         ESTABLISHED 24153/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:41212         ESTABLISHED -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58014         ESTABLISHED -
tcp        0      0 127.0.0.1:52136         127.0.0.1:12345         ESTABLISHED 24137/./client
tcp        0      1 127.0.0.1:37444         127.0.0.1:12345         SYN_SENT    24156/./client
tcp        0      0 127.0.0.1:58014         127.0.0.1:12345         ESTABLISHED 24146/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:52136         ESTABLISHED 24131/./server
```

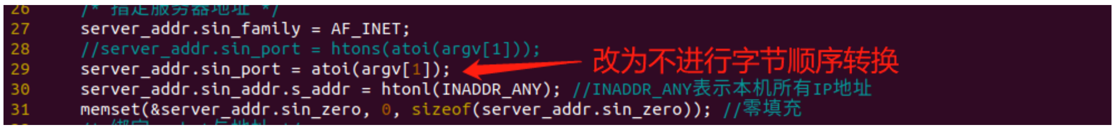
同时打开 4 个客户端，有 3 个可以被建立并正常运行

3）端口为什么要进行字节顺序转换，不转换会有什么情况？

代码改为不进行字节顺序转换：

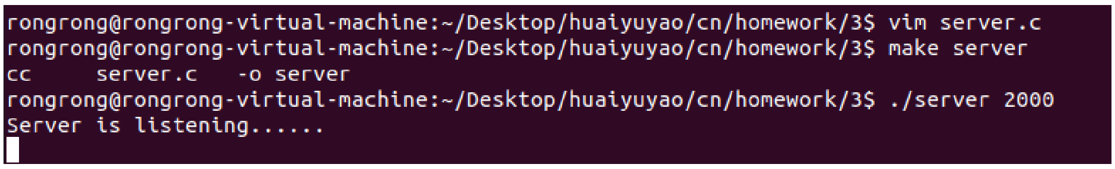
注意，这里想要看到效果，只能修改 server 端和 client 端其中一个的代码，如果两个都修改的话，相当于两个都错了，从结果来看反而是对的了。

这里我修改了 server.c 代码：

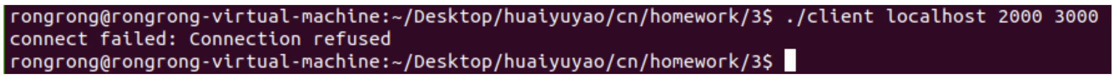


```
    /* 指定服务器地址 */
    server_addr.sin_family = AF_INET;
    //server_addr.sin_port = htons(atoi(argv[1]));
    server_addr.sin_port = atoi(argv[1]);
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY); //INADDR_ANY表示本机所有IP地址
    memset(&server_addr.sin_zero, 0, sizeof(server_addr.sin_zero)); //零填充
```

编译、运行修改后的代码，观察结果如下：



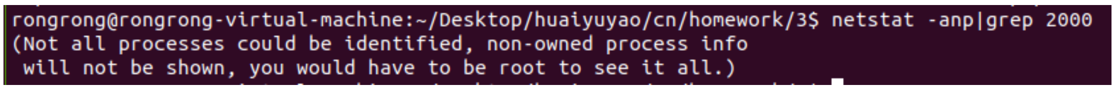
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ vim server.c
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ make server
cc     server.c   -o server
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server 2000
Server is listening......
```



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 2000 3000
connect failed: Connection refused
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

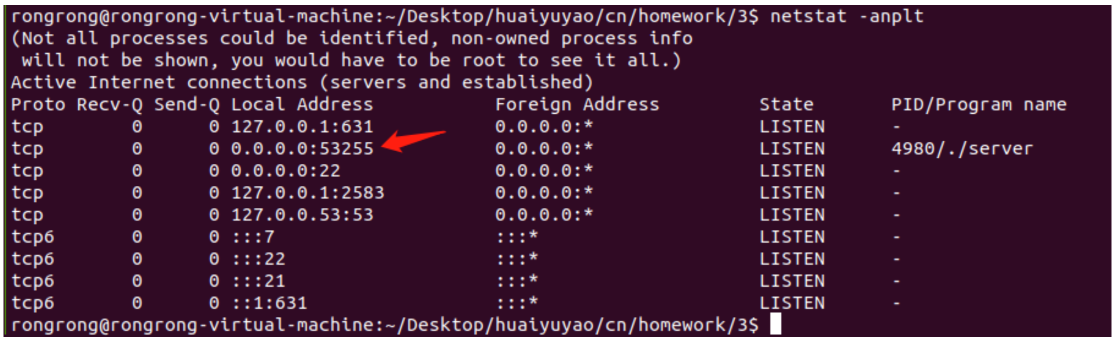
可以看到，client 端被拒绝连接了。

我们可以通过 netstat 找到答案：



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 2000
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
```

检测不到任何 2000 的端口运行。

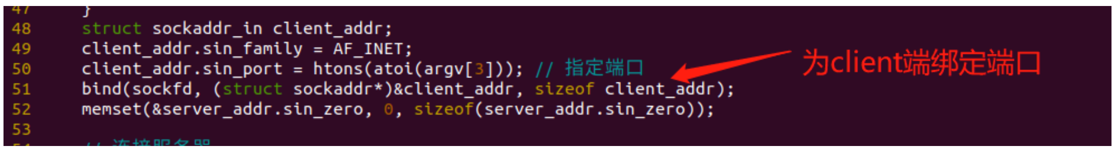


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anplt
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:53255           0.0.0.0:*               LISTEN      4980/./server
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:2583          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp6       0      0 :::7                    :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

使用`netstat -anplt`命令，我们可以看到并不是我们想要监听的 2000 端口，而是 53255，这也是为什么 client 端想要向 2000 端口发送被拒绝的原因了。

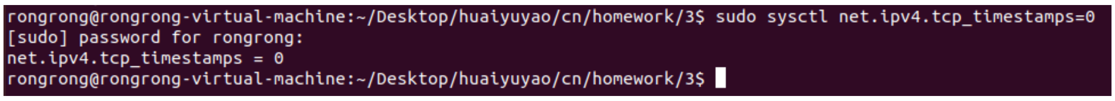
原因：在进行网络编程时，由于网络的字节顺序和主机的字节顺序可能存在不同，我们需要对它们进行转换以统一“格式”。如果没有正确对两者进行转换，从而导致两方产生了不同的解释，就会出现 bug。

4）试验客户机也像服务器一样 bind 固定端口，看看结果如何。



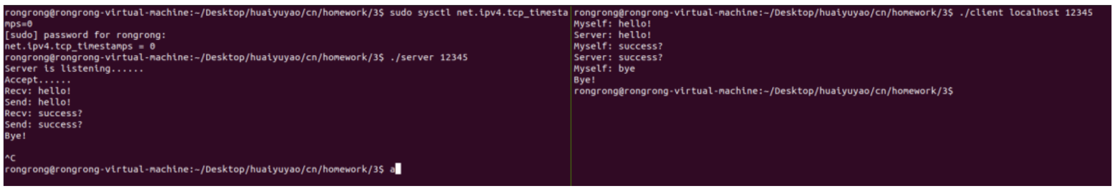
```
    struct sockaddr_in client_addr;
    client_addr.sin_family = AF_INET;
    client_addr.sin_port = htons(atoi(argv[3])); // 指定端口
    bind(sockfd, (struct sockaddr*)&client_addr, sizeof client_addr);
    memset(&server_addr.sin_zero, 0, sizeof(server_addr.sin_zero));
```

先在终端运行命令：`sudo sysctl net.ipv4.tcp_timestamps=0`



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ sudo sysctl net.ipv4.tcp_timestamps=0
[sudo] password for rongrong:
net.ipv4.tcp_timestamps = 0
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ 
```

运行客户机，连接服务器：

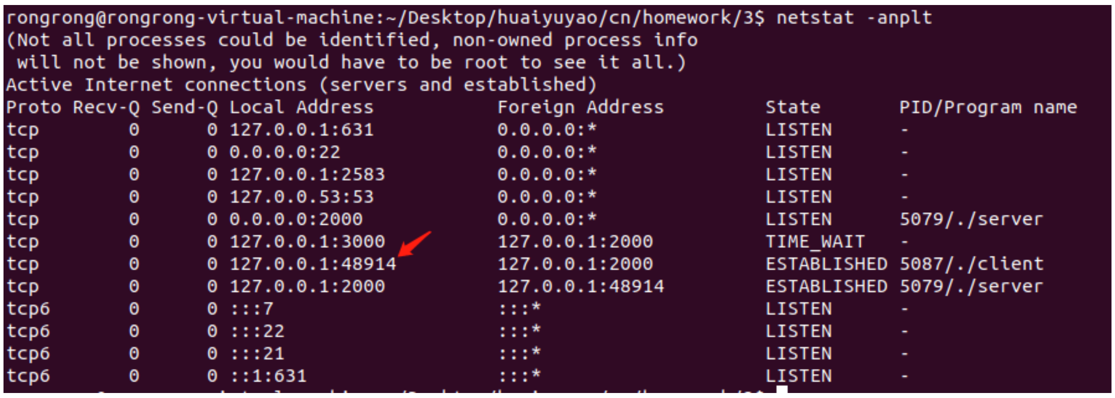


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ sudo sysctl net.ipv4.tcp_timesta
mps=0
[sudo] password for rongrong:
net.ipv4.tcp_timestamps = 0
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server 12345
Server is listening......
Accept......
Recv: hello!
Send: hello!
Recv: success?
Send: success?
Bye!

^C
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ a
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: hello!
Server: hello!
Myself: success?
Server: success?
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

观察到运行正常。

客户机主动退出、再次运行客户机，netstat 观察：

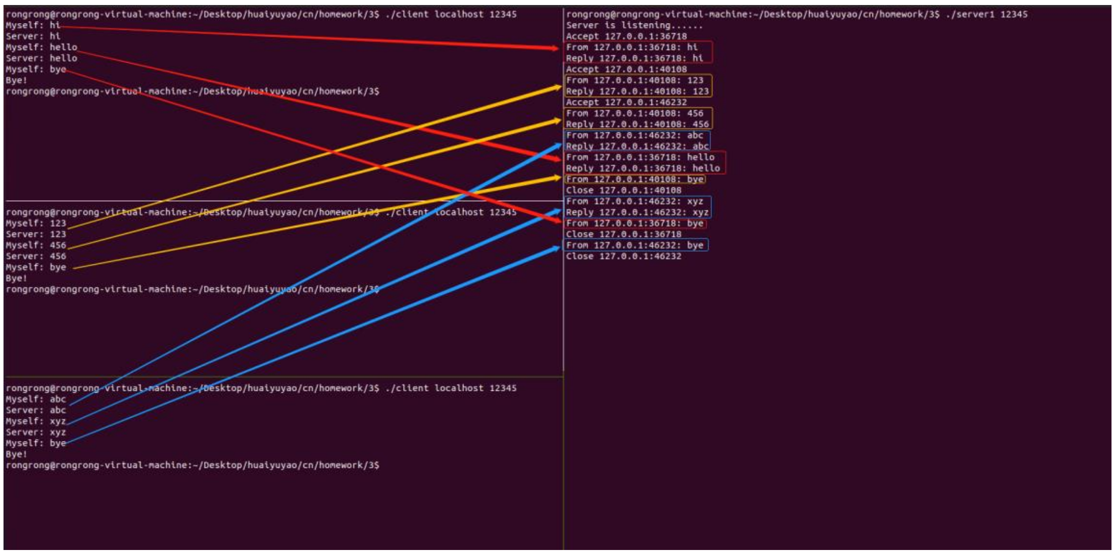


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anplt
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:2583          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:2000            0.0.0.0:*               LISTEN      5079/./server
tcp        0      0 127.0.0.1:3000          127.0.0.1:2000          TIME_WAIT   -
tcp        0      0 127.0.0.1:48914         127.0.0.1:2000          ESTABLISHED 5087/./client
tcp        0      0 127.0.0.1:2000          127.0.0.1:48914         ESTABLISHED 5079/./server
tcp6       0      0 :::7                    :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
```

使用`sudo sysctl net.ipv4.tcp_timestamps=0`命令的作用是关闭端口复用；当 3000 端口处于 TIME_WAIT 状态时，再次绑定该端口打开 client 端，系统不会将 3000 端口分配给 client 端使用，而会随机分配一个空闲状态的端口供 client 端使用。

## 3. 字符串回送服务器（TCP 并发）

1） 客户机和服务器运行时的截图



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: hi
Server: hi
Myself: hello
Server: hello
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: 123
Server: 123
Myself: 456
Server: 456
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: abc
Server: abc
Myself: xyz
Server: xyz
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

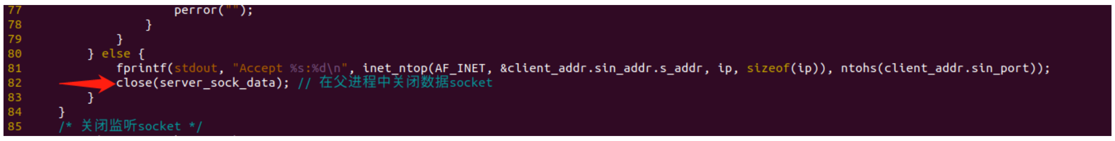
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server1 12345
Server is listening......
Accept 127.0.0.1:36718
From 127.0.0.1:36718: hi
Reply 127.0.0.1:36718: hi
Accept 127.0.0.1:40108
From 127.0.0.1:40108: 123
Reply 127.0.0.1:40108: 123
Accept 127.0.0.1:46232
From 127.0.0.1:40108: 456
Reply 127.0.0.1:40108: 456
From 127.0.0.1:46232: abc
Reply 127.0.0.1:46232: abc
From 127.0.0.1:36718: hello
Reply 127.0.0.1:36718: hello
From 127.0.0.1:40108: bye
Close 127.0.0.1:40108
From 127.0.0.1:46232: xyz
Reply 127.0.0.1:46232: xyz
From 127.0.0.1:36718: bye
Close 127.0.0.1:36718
From 127.0.0.1:46232: bye
Close 127.0.0.1:46232
```

2） 服务器 accept 之后会返回一个用于传输数据的 socket，调用 fork()会使父子进程同时拥有此 socket 描述符，父进程分支中是否需要关闭该 socket？

答：需要关闭。如果不关闭，在退出客户端后，还有多个网络处于 CLOSE_WAIT 状态。

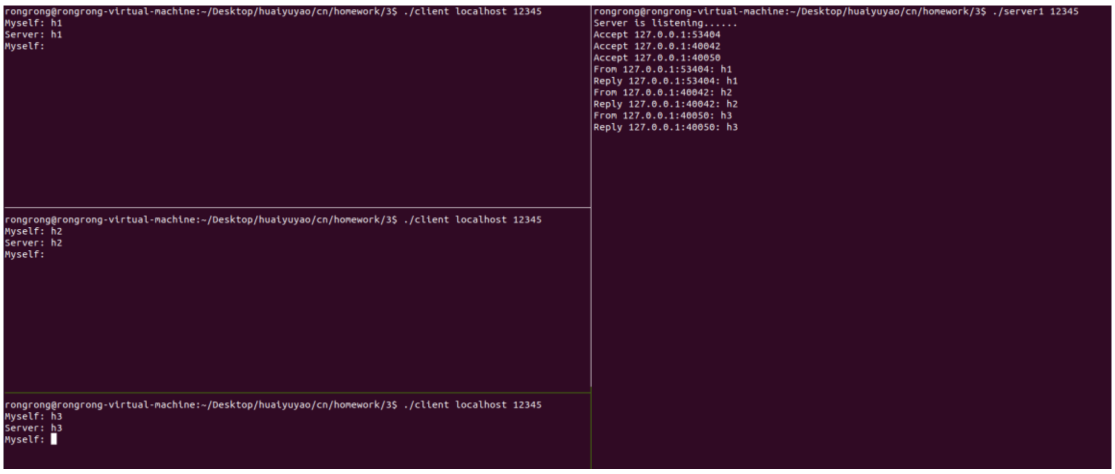
代码测试：

I.　关闭



```
                perror("");
            }
        }
    } else {
        fprintf(stdout, "Accept %s:%d\n", inet_ntop(AF_INET, &client_addr.sin_addr.s_addr, ip, sizeof(ip)), ntohs(client_addr.sin_port));
        close(server_sock_data); // 在父进程中关闭数据socket
    }
}
/* 关闭监听socket */
```

i.　查看运行是否正常



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h1
Server: h1
Myself:
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h2
Server: h2
Myself:
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h3
Server: h3
Myself: 
```

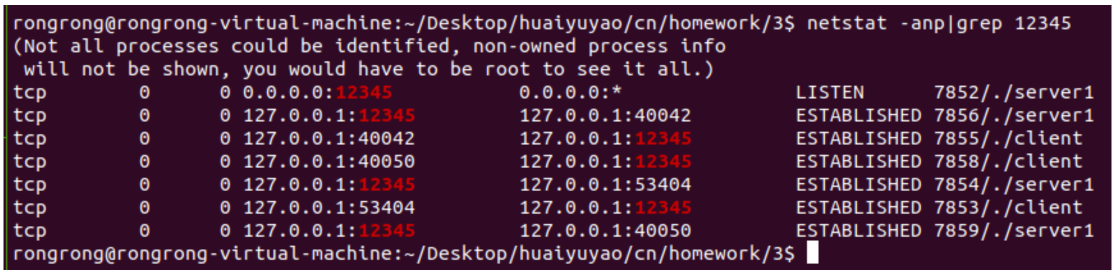
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server1 12345
Server is listening......
Accept 127.0.0.1:53404
Accept 127.0.0.1:40042
Accept 127.0.0.1:40050
From 127.0.0.1:53404: h1
Reply 127.0.0.1:53404: h1
From 127.0.0.1:40042: h2
Reply 127.0.0.1:40042: h2
From 127.0.0.1:40050: h3
Reply 127.0.0.1:40050: h3
```

可以看到，运行正常。

ii.　netstat 观察多个客户机退出后的连接状态

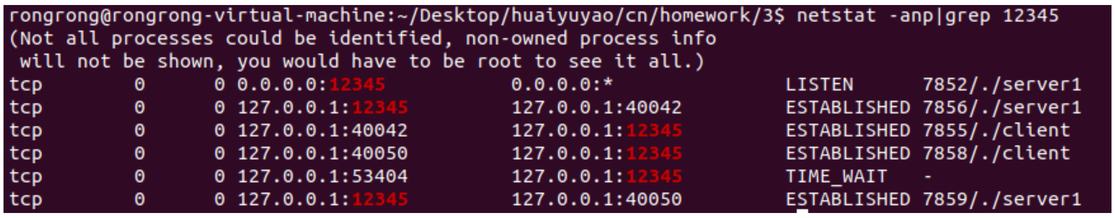
使用`netstat -anp|grep 12345`命令：

连接三个 client 端后使用该命令可以查看到如下内容：



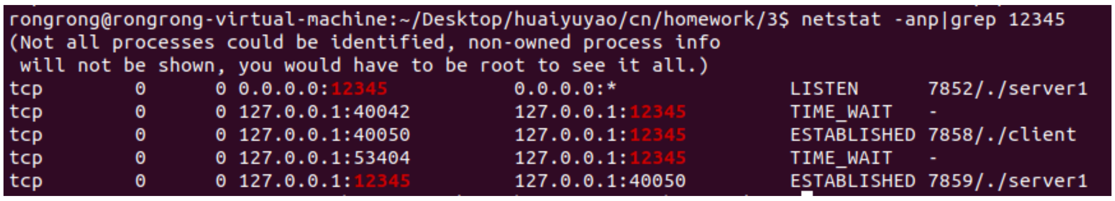
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:40042         ESTABLISHED 7856/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         ESTABLISHED 7855/./client
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         ESTABLISHED 7858/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:53404         ESTABLISHED 7854/./server1
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         ESTABLISHED 7853/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:40050         ESTABLISHED 7859/./server1
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ 
```

关闭第一个 client 端后使用该命令可以查看到如下内容：



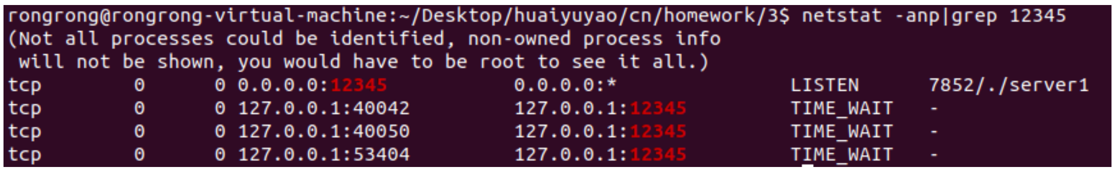
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:40042         ESTABLISHED 7856/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         ESTABLISHED 7855/./client
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         ESTABLISHED 7858/./client
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:40050         ESTABLISHED 7859/./server1
```

关闭第二个 client 端后使用该命令可以查看到如下内容：



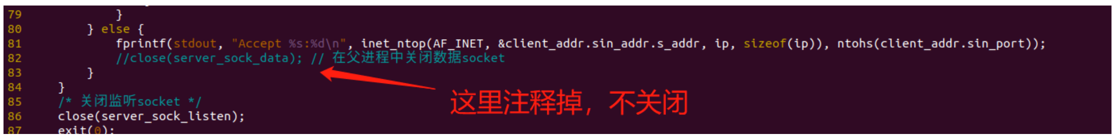
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         ESTABLISHED 7858/./client
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:40050         ESTABLISHED 7859/./server1
```

关闭最后一个 client 端后使用该命令可以查看到如下内容：



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         TIME_WAIT   -
```

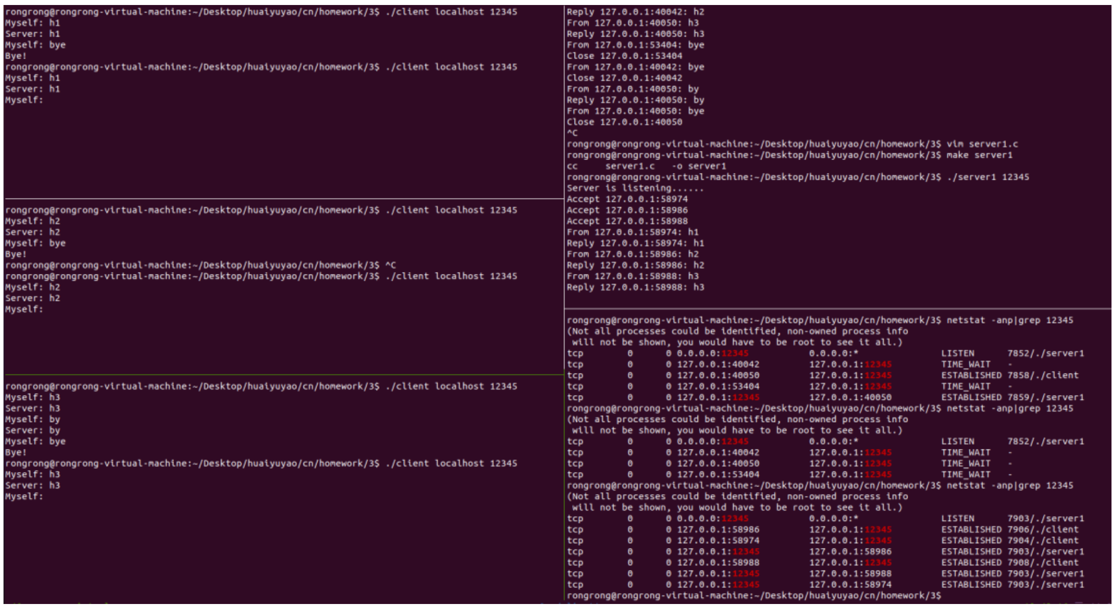
II.　　不关闭



```
            }
        } else {
            fprintf(stdout, "Accept %s:%d\n", inet_ntop(AF_INET, &client_addr.sin_addr.s_addr, ip, sizeof(ip)), ntohs(client_addr.sin_port));
            //close(server_sock_data); // 在父进程中关闭数据socket
        }
    }
    /* 关闭监听socket */
    close(server_sock_listen);
    exit(0);
```

然后重新编译。

i.　　查看运行是否正常



```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h1
Server: h1
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h1
Server: h1
Myself:
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h2
Server: h2
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ^C
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h2
Server: h2
Myself:
```

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h3
Server: h3
Myself: by
Server: by
Myself: bye
Bye!
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./client localhost 12345
Myself: h3
Server: h3
Myself:
```

```
Reply 127.0.0.1:40042: h2
From 127.0.0.1:40050: h3
Reply 127.0.0.1:40050: h3
From 127.0.0.1:53404: bye
Close 127.0.0.1:53404
From 127.0.0.1:40042: bye
Close 127.0.0.1:40042
From 127.0.0.1:40050: by
Reply 127.0.0.1:40050: by
From 127.0.0.1:40050: bye
Close 127.0.0.1:40050
^C
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ vim server1.c
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ make server1
cc     server1.c   -o server1
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server1 12345
Server is listening......
Accept 127.0.0.1:58974
Accept 127.0.0.1:58986
Accept 127.0.0.1:58988
From 127.0.0.1:58974: h1
Reply 127.0.0.1:58974: h1
From 127.0.0.1:58986: h2
Reply 127.0.0.1:58986: h2
From 127.0.0.1:58988: h3
Reply 127.0.0.1:58988: h3
```

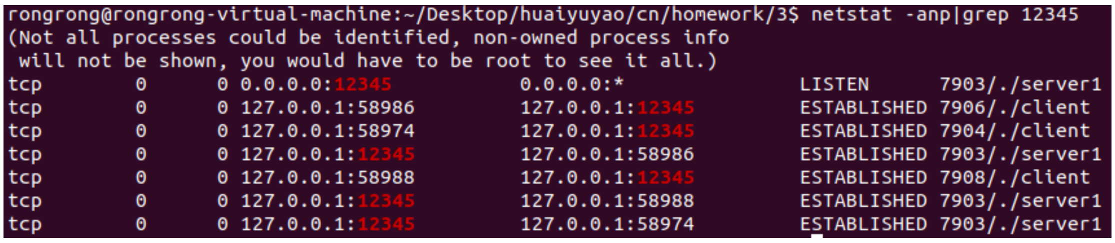
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         ESTABLISHED 7858/./client
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:40050         ESTABLISHED 7859/./server1
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7852/./server1
tcp        0      0 127.0.0.1:40042         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:40050         127.0.0.1:12345         TIME_WAIT   -
tcp        0      0 127.0.0.1:53404         127.0.0.1:12345         TIME_WAIT   -
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         ESTABLISHED 7906/./client
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         ESTABLISHED 7904/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         ESTABLISHED 7908/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         ESTABLISHED 7903/./server1
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

可以看到，不关闭此 socket 描述符，仍可以正常运行。

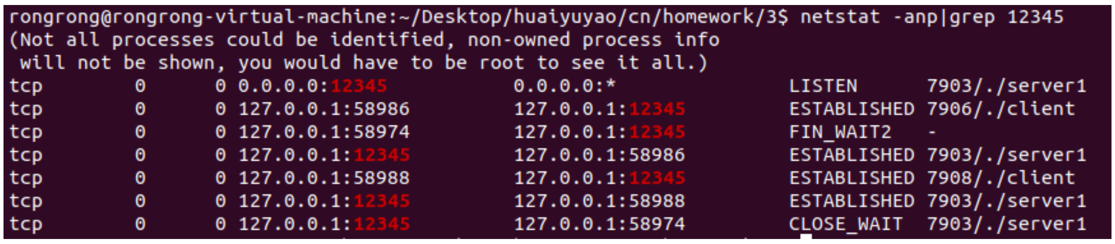
ii.　　netstat 观察多个客户机退出后的连接状态

使用`netstat -anp|grep 12345`命令：

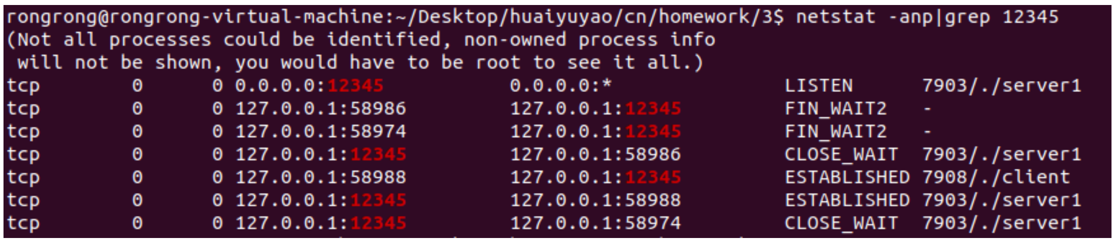
连接三个 client 端后使用该命令可以查看到如下内容：


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         ESTABLISHED 7906/./client
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         ESTABLISHED 7904/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         ESTABLISHED 7908/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         ESTABLISHED 7903/./server1
```

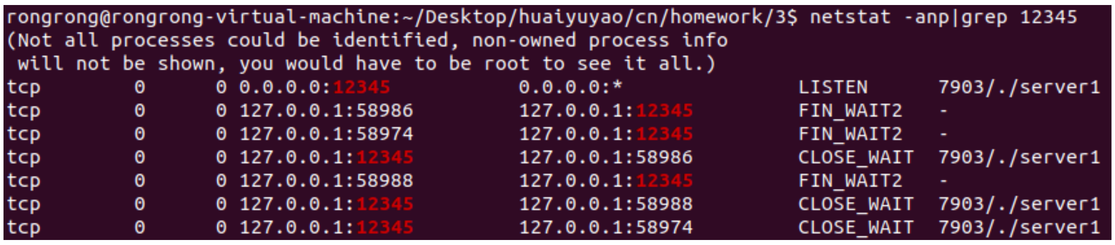
关闭第一个 client 端后使用该命令可以查看到如下内容：


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         ESTABLISHED 7906/./client
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         ESTABLISHED 7908/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         CLOSE_WAIT  7903/./server1
```

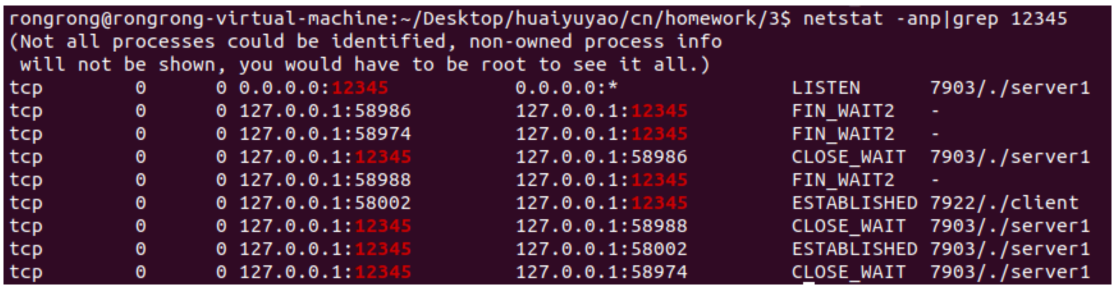
关闭第二个 client 端后使用该命令可以查看到如下内容：


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         CLOSE_WAIT  7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         ESTABLISHED 7908/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         CLOSE_WAIT  7903/./server1
```

关闭最后一个 client 端后使用该命令可以查看到如下内容：


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         CLOSE_WAIT  7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         CLOSE_WAIT  7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         CLOSE_WAIT  7903/./server1
```

若此时，再打开一个 client 端，使用该命令可以看到如下内容：



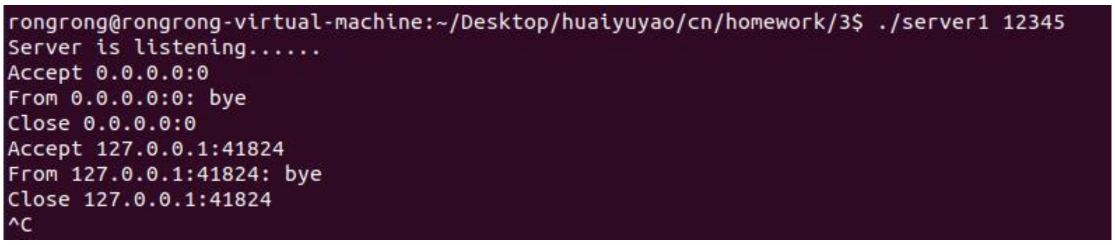
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anp|grep 12345
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7903/./server1
tcp        0      0 127.0.0.1:58986         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:58974         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:12345         127.0.0.1:58986         CLOSE_WAIT  7903/./server1
tcp        0      0 127.0.0.1:58988         127.0.0.1:12345         FIN_WAIT2   -
tcp        0      0 127.0.0.1:58002         127.0.0.1:12345         ESTABLISHED 7922/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:58988         CLOSE_WAIT  7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58002         ESTABLISHED 7903/./server1
tcp        0      0 127.0.0.1:12345         127.0.0.1:58974         CLOSE_WAIT  7903/./server1
```

之前因为和三个 client 端连接的 server 子进程，在三个 client 端关闭了之后，一直处于 CLOSE_WAIT 状态，如果之后再次有 client 端接入进来，处于 CLOSE_WAIT 状态的网络也不会自动和 client 端连接，而是会新开一个 ESTABLISHED。由于端口数量是有限的，可以预见，随着时间的推移，client 端的开关次数增多，总会到达一个时候，表面上没有多余的端口可供 client 端连接了，实际上所有的端口都处于 CLOSE_WAIT 状态，这会极大程度地造成资源利用不充分。

综上，父进程分支中应该关闭这个用于传输数据的 socket。

BUG 修复：

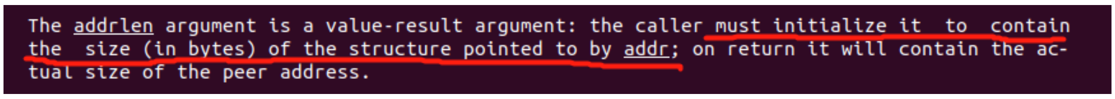
一开始完成这个程序的时候其实还有一个 bug，在第一次获取 client 端的 ip 时总是会获取到 0.0.0.0，而之后连接进来的 client 端却都是正确的。



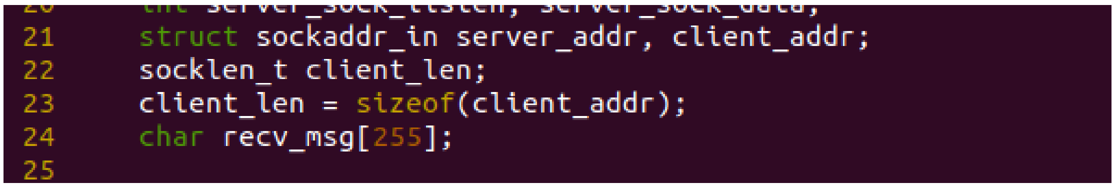
```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server1 12345
Server is listening......
Accept 0.0.0.0:0
From 0.0.0.0:0: bye
Close 0.0.0.0:0
Accept 127.0.0.1:41824
From 127.0.0.1:41824: bye
Close 127.0.0.1:41824
^C
```

为了修复这个 bug 我花费了不少工夫。原因其实是在定义变量

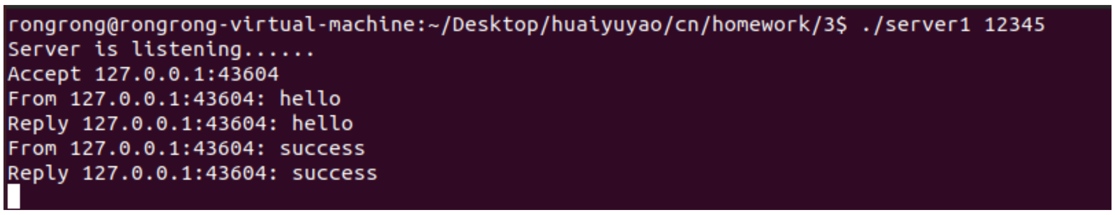
client_len 的时候，没有给它赋初始值，我翻阅 man 手册查看 accept 的函数原型发现，如果要使用 addrlen 就必须要给它赋一个初始大小：

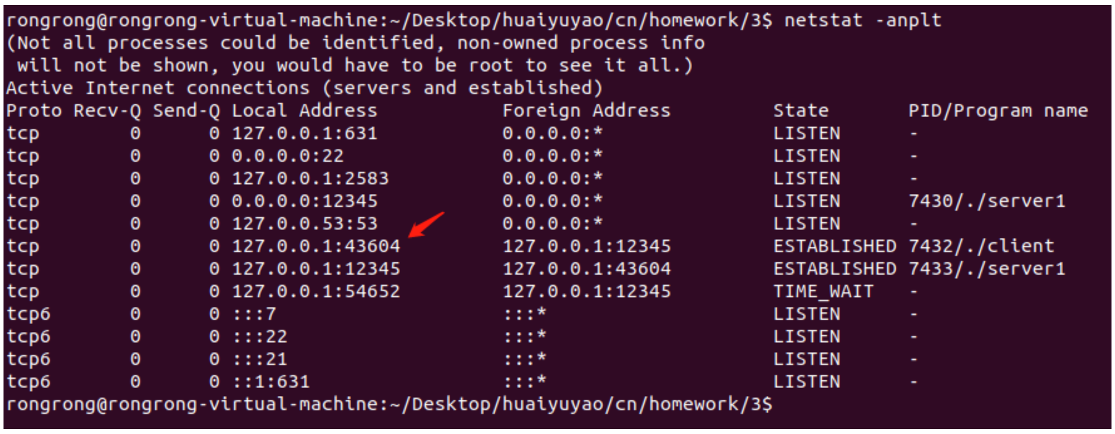

```
The addrlen argument is a value-result argument: the caller must initialize it  to  contain
the  size (in bytes) of the structure pointed to by addr; on return it will contain the ac-
tual size of the peer address.
```

于是对代码进行修改，对 client_len 赋初始值：


```
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len;
    client_len = sizeof(client_addr);
    char recv_msg[255];

```

然后再编译、运行，发现 BUG 得到了解决，可以正确得到第一个 client 端的地址和端口号：


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ ./server1 12345
Server is listening......
Accept 127.0.0.1:43604
From 127.0.0.1:43604: hello
Reply 127.0.0.1:43604: hello
From 127.0.0.1:43604: success
Reply 127.0.0.1:43604: success
```


```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$ netstat -anplt
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:2583          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      7430/./server1
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:43604         127.0.0.1:12345         ESTABLISHED 7432/./client
tcp        0      0 127.0.0.1:12345         127.0.0.1:43604         ESTABLISHED 7433/./server1
tcp        0      0 127.0.0.1:54652         127.0.0.1:12345         TIME_WAIT   -
tcp6       0      0 :::7                    :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/cn/homework/3$
```

## 三、实验结果分析

本次实验我顺利地完成了实验内容的必须要求，并且在不懈的努力下

成功修复了一个很隐蔽的bug，并完成了选做的内容。

运行./server 2000 和 ./client localhost 2000可以顺利观察到TCP的迭代情况，当第一个client端输入 “bye”终止后，server端才能接受到第二个client端发送过来的数据。

运行./server1 2000 和 ./client localhost 2000可以顺利观察到TCP的并发情况,多个client端可以几乎同时向server端发送数据，无需等待排在前面的client端终止。

## 四、实验小结与感想

本次实验完成较为顺利。

因为我提前预习了，所以在实验课前就已经将程序大致完成了。在实验课上，我着重做了选做的内容，并在洪老师的指导下，大致明白了转换字节顺序的重要性以及bind函数能实现的功能。