

Name: Haobo Yang

USC E-mail: haoboyan@usc.edu

USC ID: 6295-9882-06

Date: January 16, 2025

GitHub Repository Link: <https://github.com/HilbertYang/USC-EE533-projects>

Youtube Link: [EE533 lab1 HaoboYang](#)

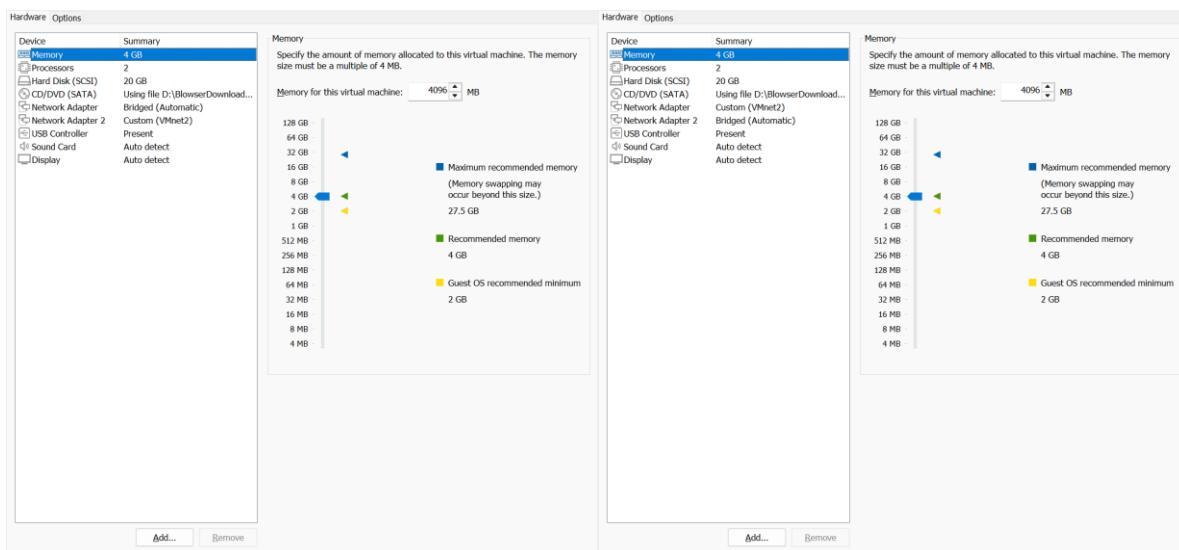
EE533 Lab1 Report

Introduction

In this lab, a client–server communication pattern was implemented using sockets in a Linux environment. Two virtual machines were configured on VMware to enable network communication between a client VM and a server VM. Sample code was used to test message sending, receiving, and replying over a TCP connection. The server was further modified to support multiple client connections, as well as UNIX domain sockets and single-process concurrent servers. This lab helped develop a practical understanding of basic socket programming and client–server communication.

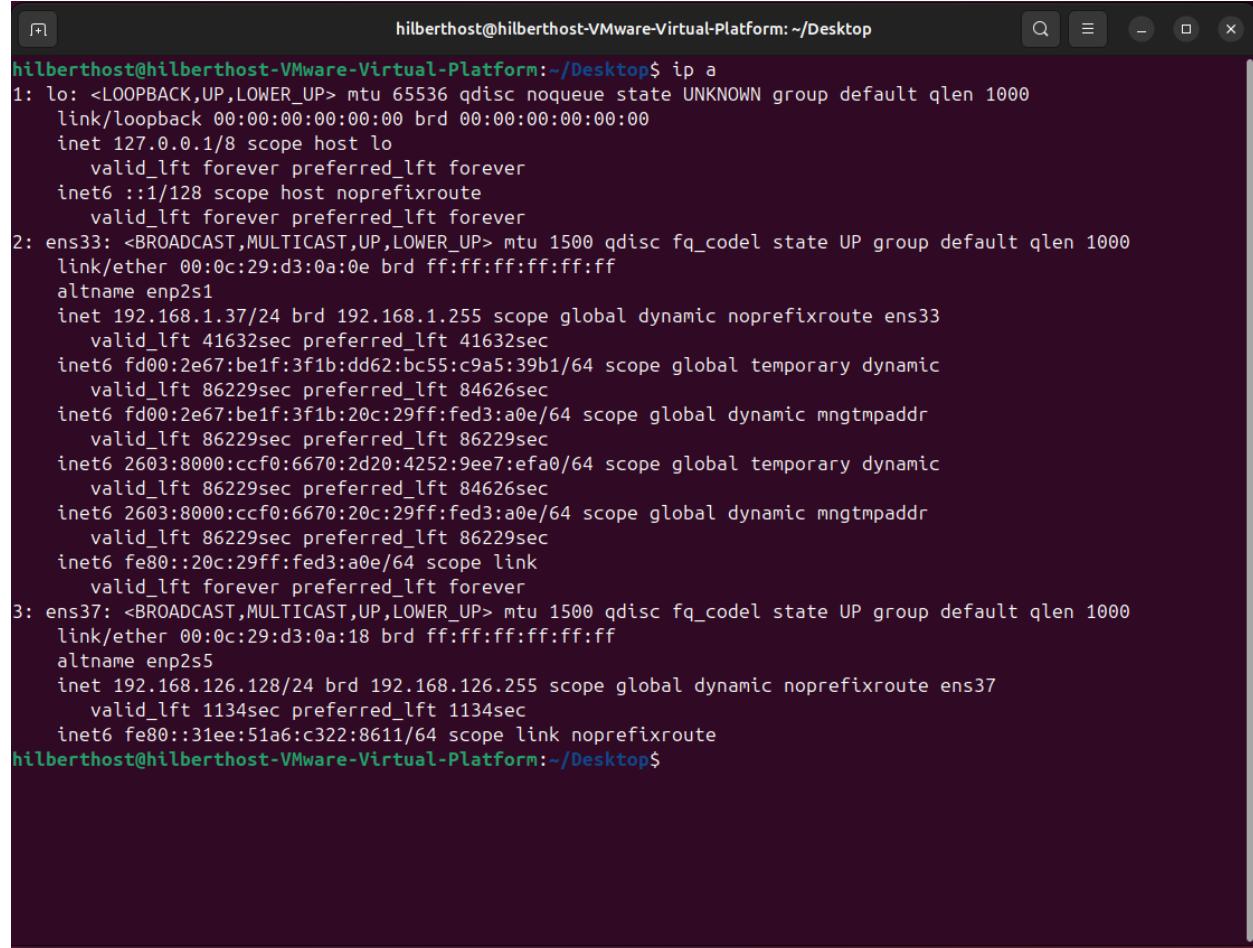
Environment establish

VM Network Setting



The VM1 and VM2 were set to connect bridge network adapter. By doing this, both of them can connect to the internet through the host computer.

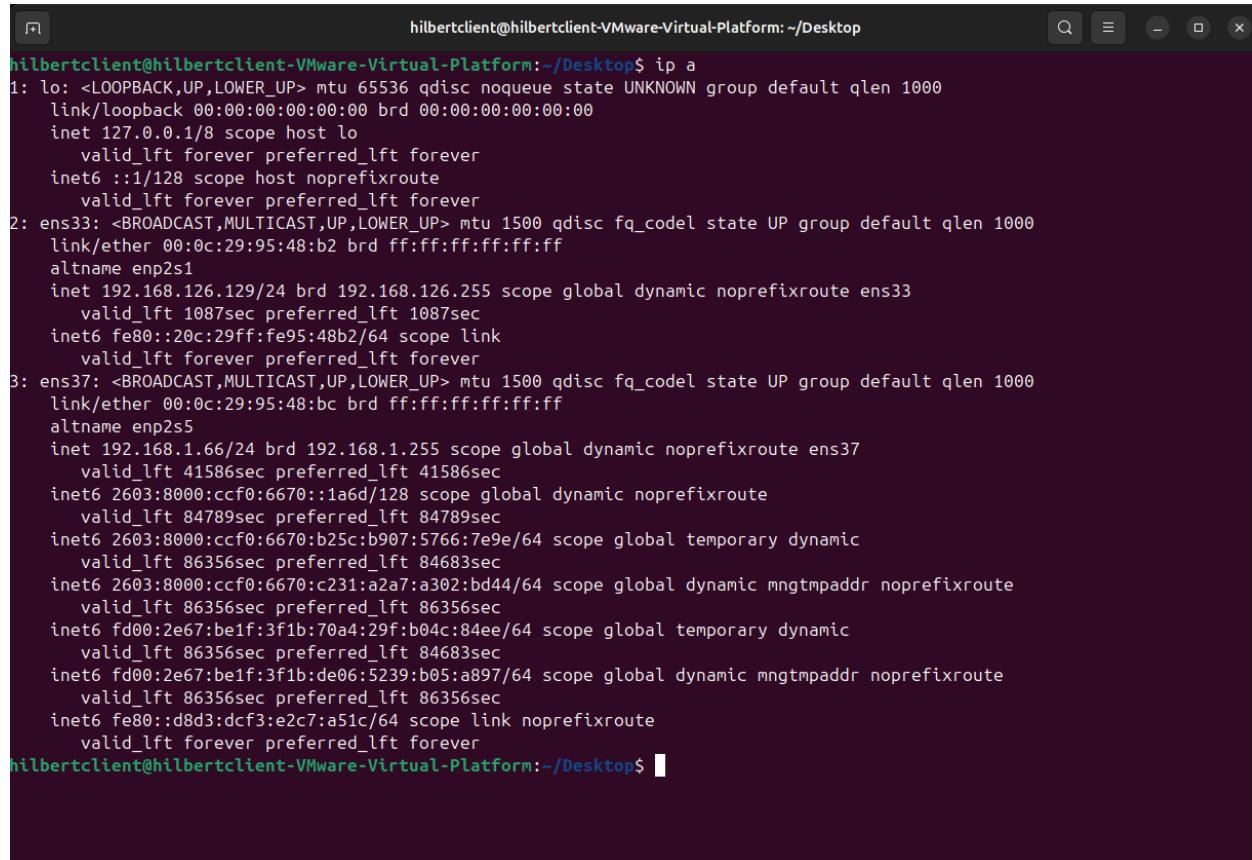
Besides that, VM1 and VM2 were also set into a Vnet2, a local network between VMs.



A screenshot of a terminal window titled "hilberthost@hilberthost-VMware-Virtual-Platform: ~/Desktop". The window contains the output of the command "ip a". The output shows three network interfaces: "lo" (loopback), "ens33" (ethernet), and "ens37" (ethernet). The "lo" interface has an IP address of 127.0.0.1. The "ens33" interface has an IP address of 192.168.1.255. The "ens37" interface has an IP address of 192.168.126.255. The output includes details about MTU, queueing discipline (qdisc), link layer information, and various IP configurations (inet, inet6) with their respective subnet masks, broadcast addresses, lifetimes, and lifetimes.

```
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d3:0a:0e brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.255/24 brd 192.168.1.255 scope global dynamic noprefixroute ens33
        valid_lft 41632sec preferred_lft 41632sec
    inet6 fd00:2e67:be1f:3f1b:dd62:bc55:c9a5:39b1/64 scope global temporary dynamic
        valid_lft 86229sec preferred_lft 84626sec
    inet6 fd00:2e67:be1f:3f1b:20c:29ff:fed3:a0e/64 scope global dynamic mngtmpaddr
        valid_lft 86229sec preferred_lft 86229sec
    inet6 2603:8000:ccf0:6670:2d20:4252:9ee7:efa0/64 scope global temporary dynamic
        valid_lft 86229sec preferred_lft 84626sec
    inet6 2603:8000:ccf0:6670:20c:29ff:fed3:a0e/64 scope global dynamic mngtmpaddr
        valid_lft 86229sec preferred_lft 86229sec
    inet6 fe80::20c:29ff:fed3:a0e/64 scope link
        valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d3:0a:18 brd ff:ff:ff:ff:ff:ff
    altname enp2s5
    inet 192.168.126.128/24 brd 192.168.126.255 scope global dynamic noprefixroute ens37
        valid_lft 1134sec preferred_lft 1134sec
    inet6 fe80::31ee:51a6:c322:8611/64 scope link noprefixroute
```

hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop\$



A screenshot of a terminal window titled "hilbertclient@hilbertclient-VMware-Virtual-Platform: ~/Desktop". The window displays the output of the command "ip a". The output shows three network interfaces: "lo" (loopback), "ens33" (ethernet), and "ens37" (ethernet). Each interface has one or more inet (IPv4) and/or inet6 (IPv6) entries. The interfaces are in a UP state with MTU values ranging from 1500 to 65536. Quality of Service (qdisc) disciplines like "fq_codel" and "noqueue" are applied. Link layer information like link layer address (MAC address) and broadcast address is also provided.

```
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:95:48:b2 brd ff:ff:ff:ff:ff:ff
        altname enp2s1
        inet 192.168.126.129/24 brd 192.168.126.255 scope global dynamic noprefixroute ens33
            valid_lft 1087sec preferred_lft 1087sec
        inet6 fe80::20c:29ff:fe95:48b2/64 scope link
            valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:95:48:bc brd ff:ff:ff:ff:ff:ff
        altname enp2s5
        inet 192.168.1.66/24 brd 192.168.1.255 scope global dynamic noprefixroute ens37
            valid_lft 41586sec preferred_lft 41586sec
        inet6 2603:8000:ccf0:6670::1a6d/128 scope global dynamic noprefixroute
            valid_lft 84789sec preferred_lft 84789sec
        inet6 2603:8000:ccf0:6670:b25c:b907:5766:7e9e/64 scope global temporary dynamic
            valid_lft 86356sec preferred_lft 84683sec
        inet6 2603:8000:ccf0:6670:c231:a2a7:a302:bd44/64 scope global dynamic mngtmpaddr noprefixroute
            valid_lft 86356sec preferred_lft 86356sec
        inet6 fd00:2e67:be1f:3f1b:70a4:29f:b04c:84ee/64 scope global temporary dynamic
            valid_lft 86356sec preferred_lft 84683sec
        inet6 fd00:2e67:be1f:3f1b:de06:5239:b05:a897/64 scope global dynamic mngtmpaddr noprefixroute
            valid_lft 86356sec preferred_lft 86356sec
        inet6 fe80::d8d3:dcf3:e2c7:a51c/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$
```

Client-server communication

Sample Test

When doing TCP communication, the server use bind(), listen(), accept(), and read()/write().

The client uses the gethostbyname() function to retrieve the IP address from the DNS and uses bcopy to copy the information into the serv_addr. It then establishes the TCP connection using the connect function and performs read()/write() operations.

```

server.c:25:6: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
  25 |     bzero((char *) &serv_addr, sizeof(serv_addr));
      |     ^
server.c:25:6: warning: incompatible implicit declaration of built-in function 'bzero' [-Wbuiltin-declaration-mismatch]
server.c:26:15: warning: implicit declaration of function 'atoi' [-Wimplicit-function-declaration]
  26 |     portno = atoi(argv[1]);
      |     ^
server.c:39:10: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
  39 |     n = read(newsockfd,buffer,255);
      |     ^
      |     fread
server.c:42:10: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
  42 |     n = write(newsockfd,"I got your message",18);
      |     ^
      |     fwrite
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server 5000
Here is the message: i love ee533
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ 

hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 
client.c:45:29: warning: implicit declaration of function 'fwrite' [-Wimplicit-function-declaration]
  45 |         fwrite
      |         ^
client.c:45:29: warning: implicit declaration of function 'strlen' [-Wimplicit-function-declaration]
  45 |             n = write(sockfd,buffer,strlen(buffer));
      |             ^
client.c:6:1: note: include '<string.h>' or provide a declaration of 'strlen'
  5 | #include <netdb.h>
      | ^~~~~
client.c:6:1: note: include '<string.h>' or provide a declaration of 'strlen'
  6 |
client.c:45:29: warning: incompatible implicit declaration of built-in function 'strlen' [-Wbuiltin-declaration-mismatch]
  45 |             n = write(sockfd,buffer,strlen(buffer));
      |             ^
client.c:45:29: note: include '<string.h>' or provide a declaration of 'strlen'
client.c:49:9: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
  49 |     n = read(sockfd,buffer,255);
      |     ^
      |     fread
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5000
Please enter the message: i love ee533
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 

```

Adding Process-Based Concurrency Using fork()

By using a child process, the server can just use the main function to listen to the channel and keep accepting new clients. When it accepts a new client, the host forks a new child process do stuff (), which has read()/write() inside.

```

client.c:45:29: warning: implicit declaration of function 'fwrite' [-Wimplicit-function-declaration]
  45 |         fwrite
      |         ^
client.c:45:29: warning: implicit declaration of function 'strlen' [-Wimplicit-function-declaration]
  45 |             n = write(sockfd,buffer,strlen(buffer));
      |             ^
client.c:6:1: note: include '<string.h>' or provide a declaration of 'strlen'
  5 | #include <netdb.h>
      | ^~~~~
client.c:6:1: note: include '<string.h>' or provide a declaration of 'strlen'
  6 |
client.c:45:29: warning: incompatible implicit declaration of built-in function 'strlen' [-Wbuiltin-declaration-mismatch]
  45 |             n = write(sockfd,buffer,strlen(buffer));
      |             ^
client.c:45:29: note: include '<string.h>' or provide a declaration of 'strlen'
client.c:49:9: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
  49 |     n = read(sockfd,buffer,255);
      |     ^
      |     fread
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5000
Please enter the message: hi!
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 

hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc enhanced_server.c -o server
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server 5000
Here is the message: hello!
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ 

```

UDP

In the UDP protocol, the connections between the server and client are not needed anymore.

Instead of connection(), read(), and write, UDP protocol use sendto() and recvfrom()

```

hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc server_udp.c -o server
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server 5000
Here is the message: hello world
^C
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ 

hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ gcc client_udp.c -o client
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5000
Please enter the message: hello world
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 

```

UNIX Domain Socket Implementation

Using the UNIX Domain socket, we can directly use the file path as the address of the server

```

hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc U_server.c -o server
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc U_client.c -o client
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server
Unix domain server listening on /tmp/ee533_unix_sock
Got: hello goodmorning
^C
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server
Unix domain server listening on /tmp/ee533_unix_sock
Got: hello good morning!
^C

hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./client "hello good morning!"
Reply: I got your message
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ 

```

Single-process Concurrent Server

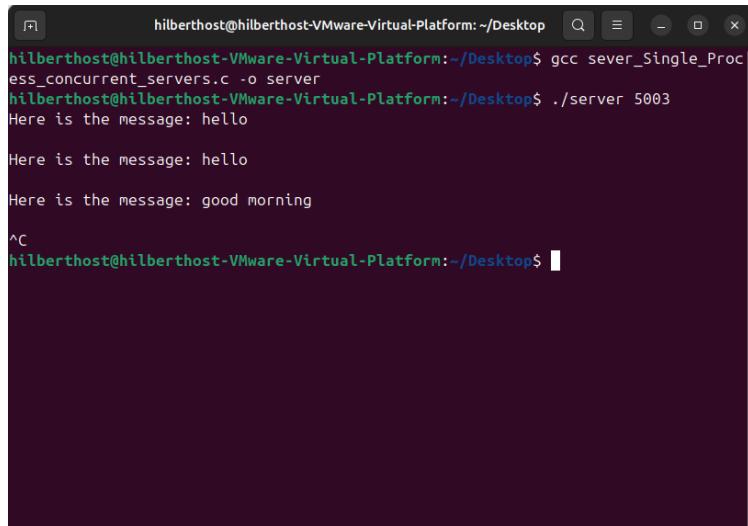
In a single-process concurrent server, there is no fork. Server use only one process to catch the connection. However, it does it in kind of selection.

```

hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5003
Please enter the message: hello
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5003
Please enter the message: good morning
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 

hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ ./client 192.168.126.128 5003
Please enter the message: hello
I got your message
hilbertclient@hilbertclient-VMware-Virtual-Platform:~/Desktop$ 

```



```
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc sever_Single_Proc  
cess_concurrent_servers.c -o server  
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server 5003  
Here is the message: hello  
Here is the message: hello  
Here is the message: good morning  
^C  
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$
```