# EE533

## LAB 1

CREATED AND COMPILED BY:

SARTHAK JAIN

(UNIVERSITY OF SOUTHERN CALIFORNIA)

***GITHUB LINK FOR MY REPOSITORY:***

You'll find all the codes and the executables on this repository.

https://github.com/SARTHAK-JAIN-ASIC/EE533/tree/main/LAB1

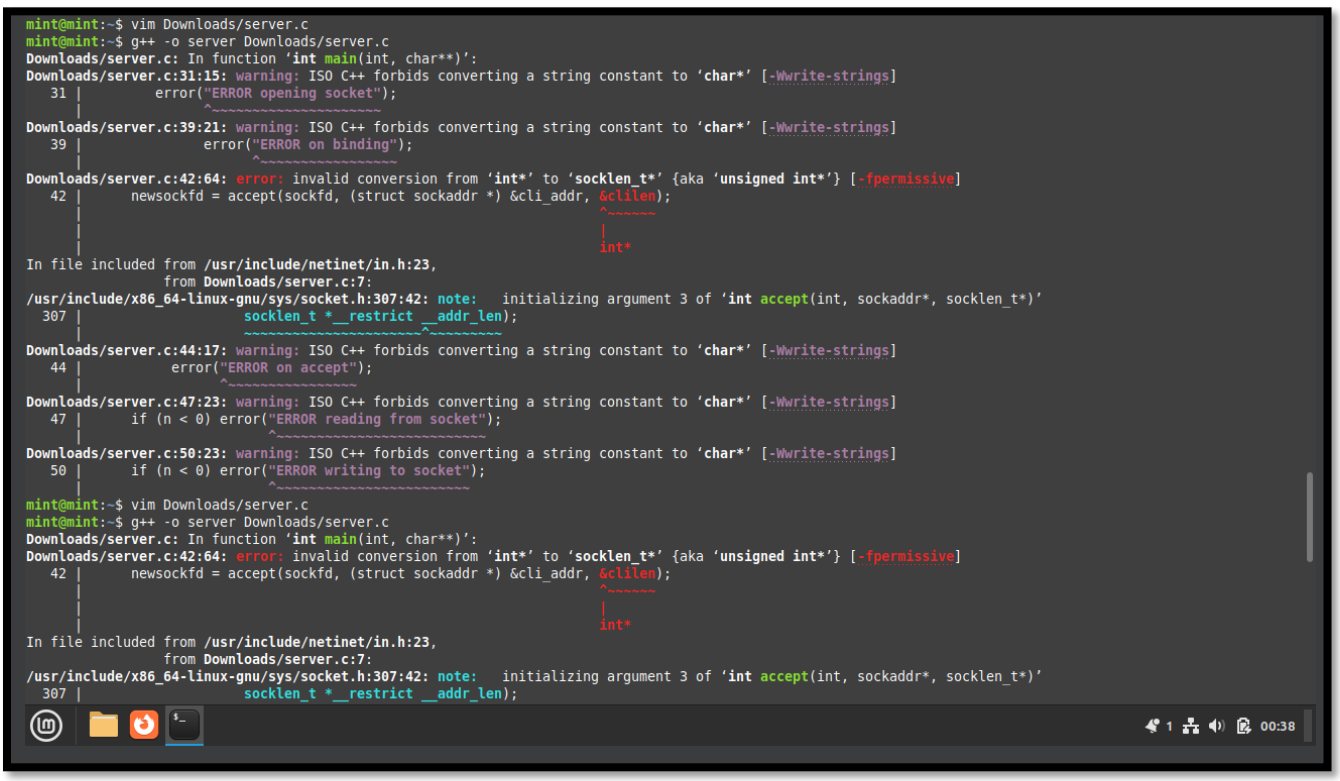
**PART 1:**

In the first part, we first had to make changes to the original code provided.

1. We added a few header files (the lack of which was causing compilation errors on my end).
2. We changed the error function's arguments to be const instead of variable since we cannot convert a constant string to char*
3. Changed data type of clilen from int to socklen_t.

```
mint@mint:~/Downloads$ vim client.c
mint@mint:~/Downloads$ g++ -o server client.c
client.c:11:6: error: variable or field ‘error’ declared void
   11 | void error(cosnt char *msg)
      |      ^~~~~
client.c:11:12: error: ‘cosnt’ was not declared in this scope; did you mean ‘const’?
   11 | void error(cosnt char *msg)
      |            ^~~~~
      |            const
client.c: In function ‘int main(int, char**)’:
client.c:32:9: error: ‘error’ was not declared in this scope; did you mean ‘perror’?
   32 |         error("ERROR opening socket");
      |         ^~~~~
      |         perror
client.c:45:9: error: ‘error’ was not declared in this scope; did you mean ‘perror’?
   45 |         error("ERROR connecting");
      |         ^~~~~
      |         perror
client.c:51:10: error: ‘error’ was not declared in this scope; did you mean ‘perror’?
   51 |          error("ERROR writing to socket");
      |          ^~~~~
      |          perror
client.c:55:10: error: ‘error’ was not declared in this scope; did you mean ‘perror’?
   55 |          error("ERROR reading from socket");
      |          ^~~~~
      |          perror
mint@mint:~/Downloads$ vim client.c
mint@mint:~/Downloads$ g++ -o server client.c
```

```
mint@mint:~$ vim Downloads/server.c
mint@mint:~$ g++ -o server Downloads/server.c
Downloads/server.c: In function ‘int main(int, char**)’:
Downloads/server.c:31:15: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   31 |         error("ERROR opening socket");
      |               ^~~~~~~~~~~~~~~~~~~~~~
Downloads/server.c:39:21: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   39 |              error("ERROR on binding");
      |                     ^~~~~~~~~~~~~~~~~
Downloads/server.c:42:64: error: invalid conversion from ‘int*’ to ‘socklen_t*’ {aka ‘unsigned int*’} [-fpermissive]
   42 |      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
      |                                                                ^~~~~~~
      |                                                                |
      |                                                                int*
In file included from /usr/include/netinet/in.h:23,
                 from Downloads/server.c:7:
/usr/include/x86_64-linux-gnu/sys/socket.h:307:42: note:   initializing argument 3 of ‘int accept(int, sockaddr*, socklen_t*)’
  307 |                    socklen_t *__restrict __addr_len);
      |                    ~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~
Downloads/server.c:44:17: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   44 |          error("ERROR on accept");
      |                ^~~~~~~~~~~~~~~~~
Downloads/server.c:47:23: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   47 |      if (n < 0) error("ERROR reading from socket");
      |                       ^~~~~~~~~~~~~~~~~~~~~~~~~~
Downloads/server.c:50:23: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   50 |      if (n < 0) error("ERROR writing to socket");
      |                       ^~~~~~~~~~~~~~~~~~~~~~~~~~
mint@mint:~$ vim Downloads/server.c
mint@mint:~$ g++ -o server Downloads/server.c
Downloads/server.c: In function ‘int main(int, char**)’:
Downloads/server.c:42:64: error: invalid conversion from ‘int*’ to ‘socklen_t*’ {aka ‘unsigned int*’} [-fpermissive]
   42 |      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
      |                                                                ^~~~~~~
      |                                                                |
      |                                                                int*
In file included from /usr/include/netinet/in.h:23,
                 from Downloads/server.c:7:
/usr/include/x86_64-linux-gnu/sys/socket.h:307:42: note:   initializing argument 3 of ‘int accept(int, sockaddr*, socklen_t*)’
  307 |                    socklen_t *__restrict __addr_len);
```



mint@mint: ~/Downloads

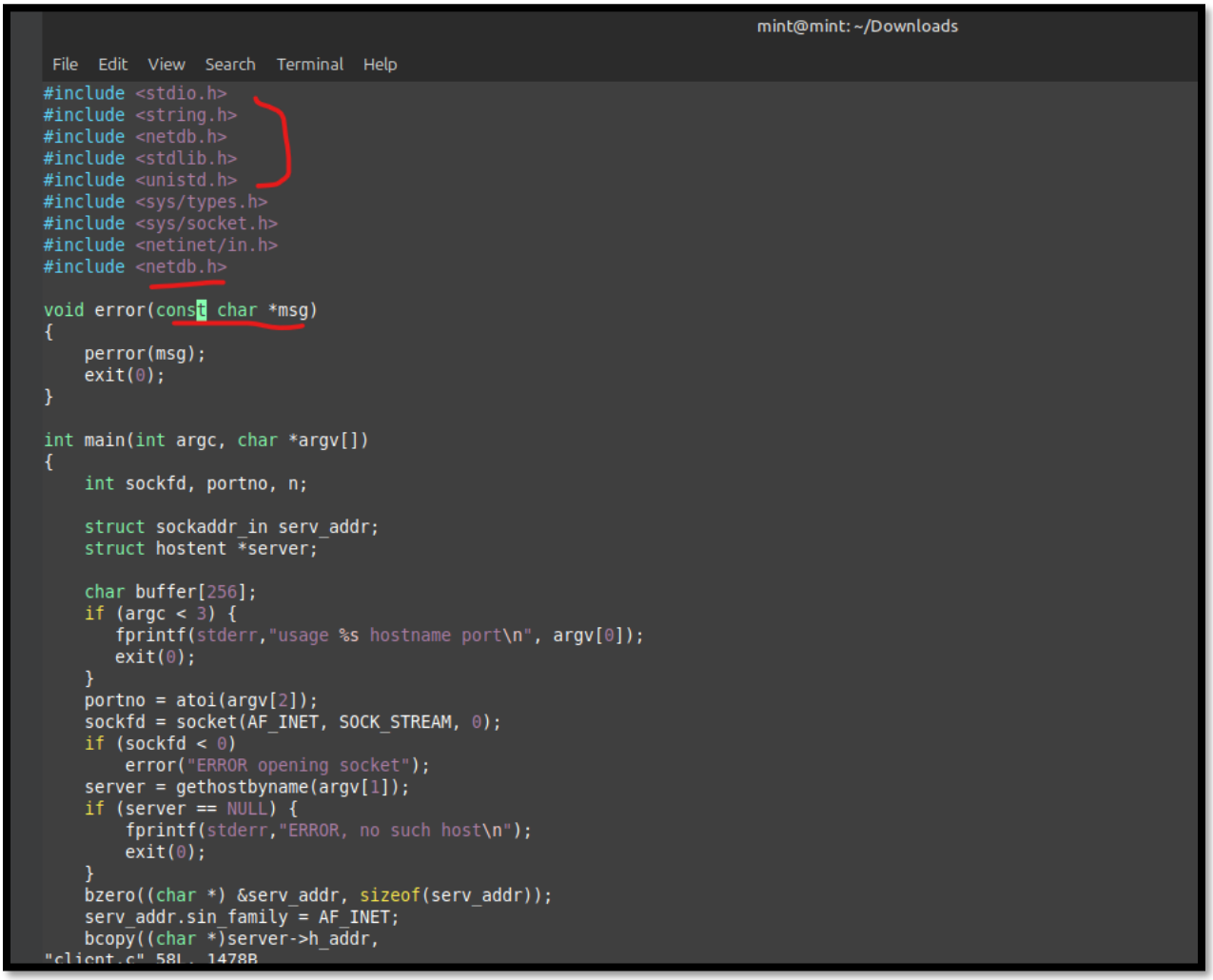
File Edit View Search Terminal Help

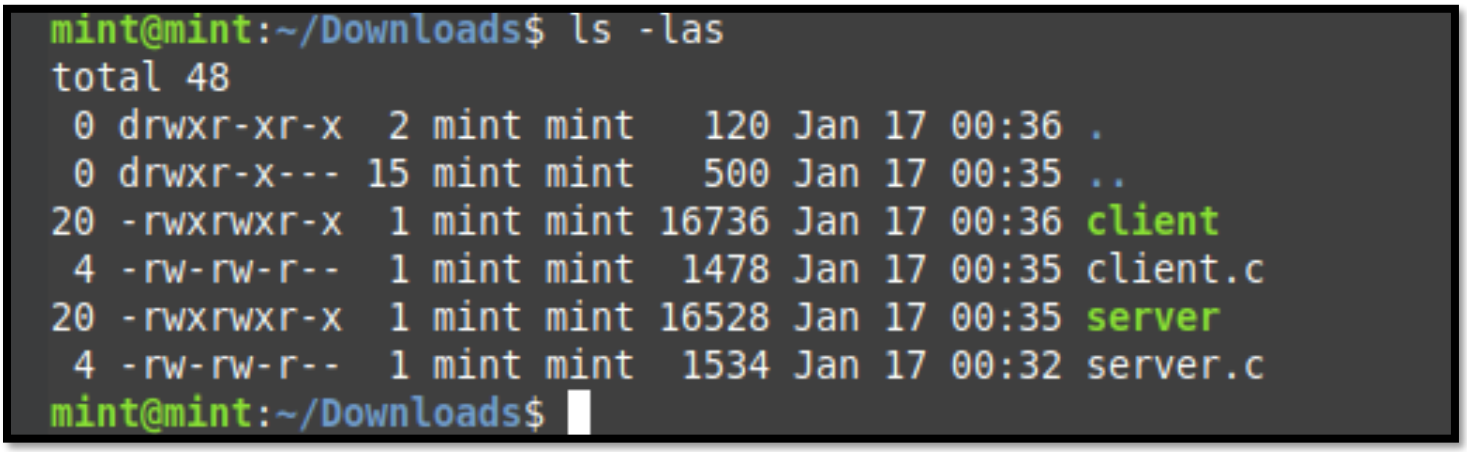
```c
/* A simple server in the internet domain using TCP
   The port number is passed as an argument */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>


void error(const char* msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno;// clilen;
     socklen_t clilen;
     char buffer[256];
     struct sockaddr_in serv_addr, cli_addr;
     int n;
     if (argc < 2) {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");
     bzero((char *) &serv_addr, sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
```

mint@mint: ~/Downloads

File Edit View Search Terminal Help

```c
/* A simple server in the internet domain using TCP
   The port number is passed as an argument */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>


void error(const char* msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno;// clilen;
     socklen_t clilen;
     char buffer[256];
     struct sockaddr_in serv_addr, cli_addr;
     int n;
     if (argc < 2) {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");
     bzero((char *) &serv_addr, sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
```



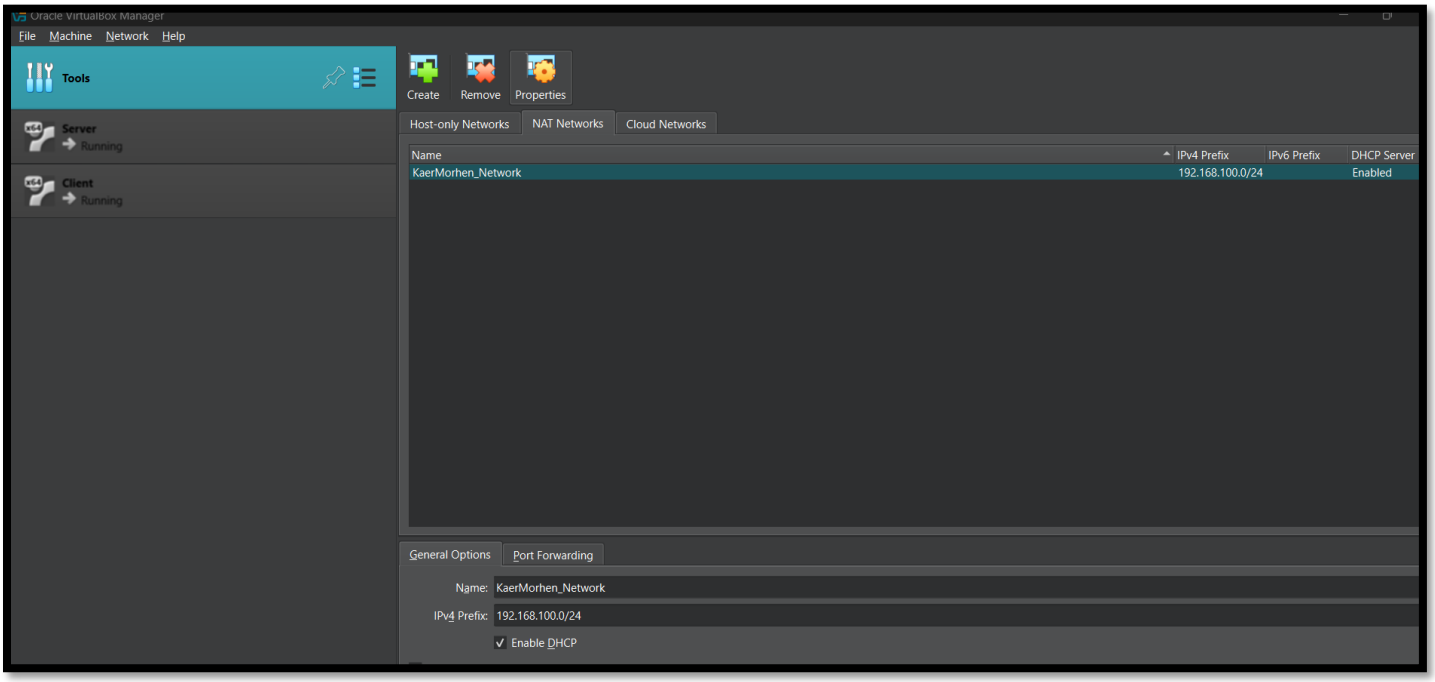




***All these changes enabled us to compile the code correctly as seen above.***

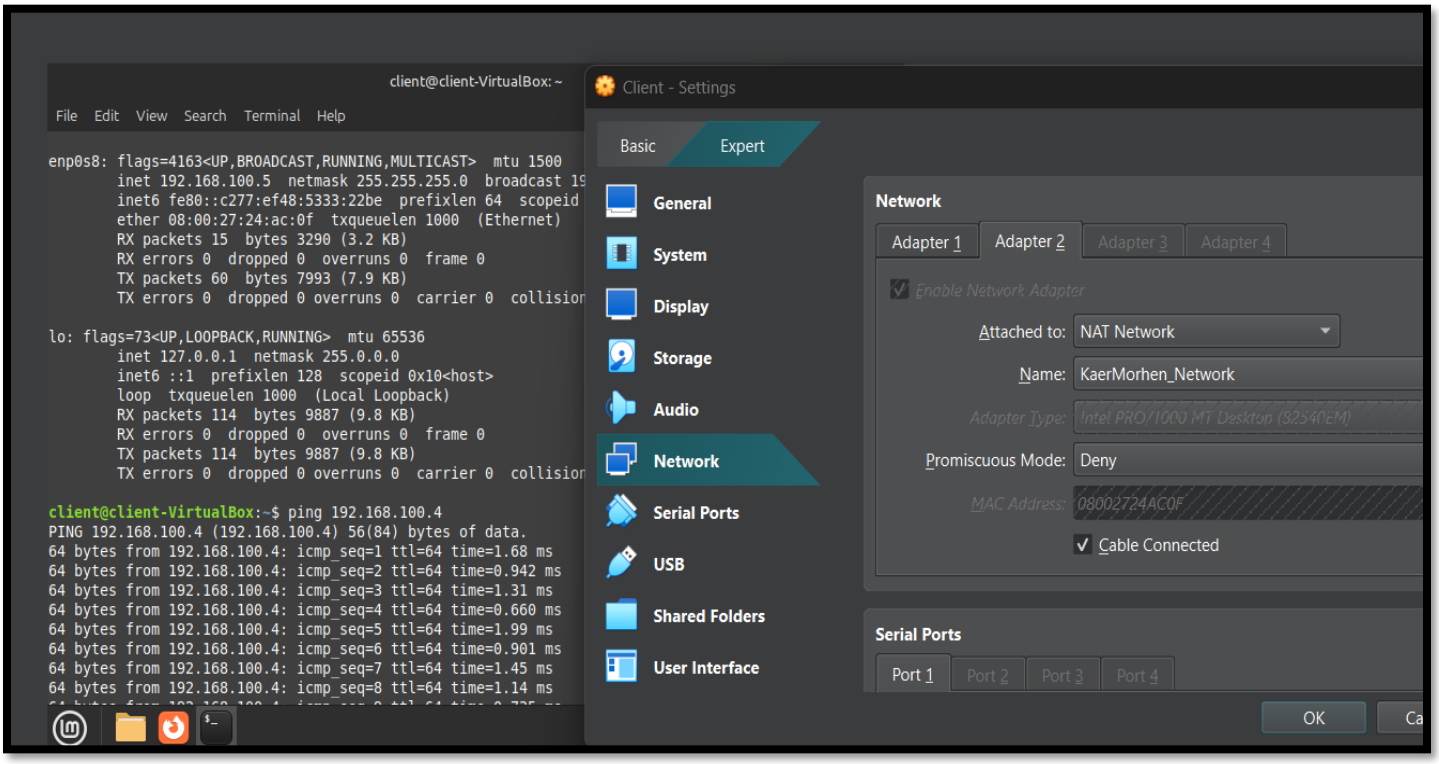
**PART 2:**

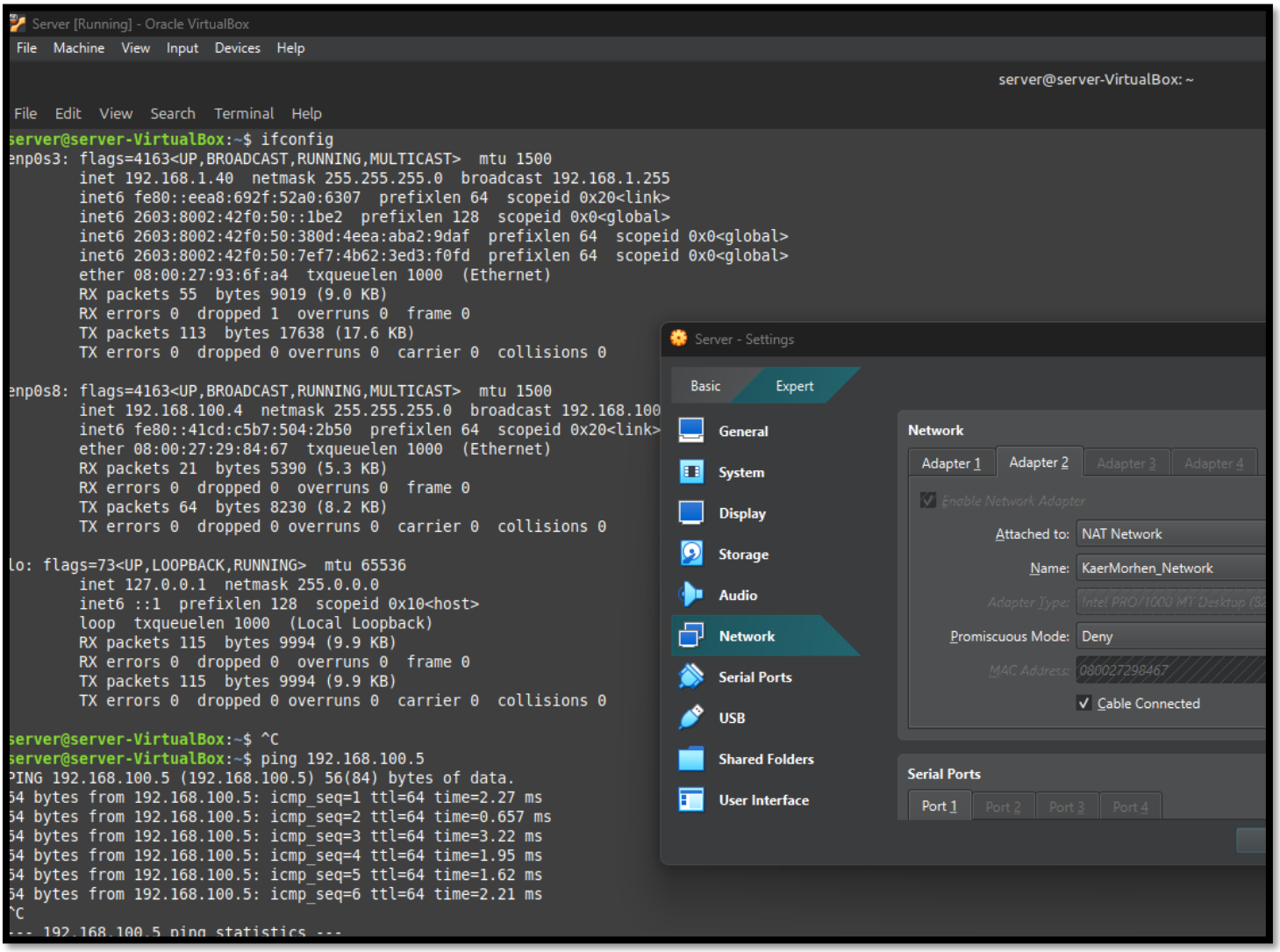
Next task was to get the client and the server to ping each other.

For this, I created a NAT network, that would “connect” server and client VMs (as below)

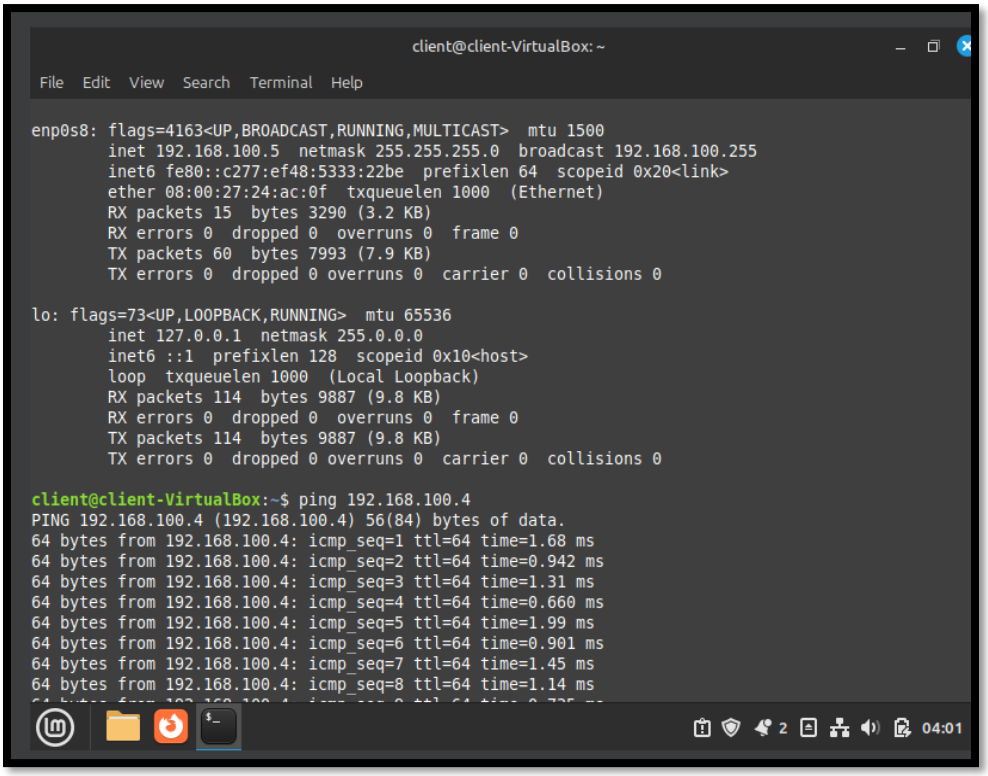


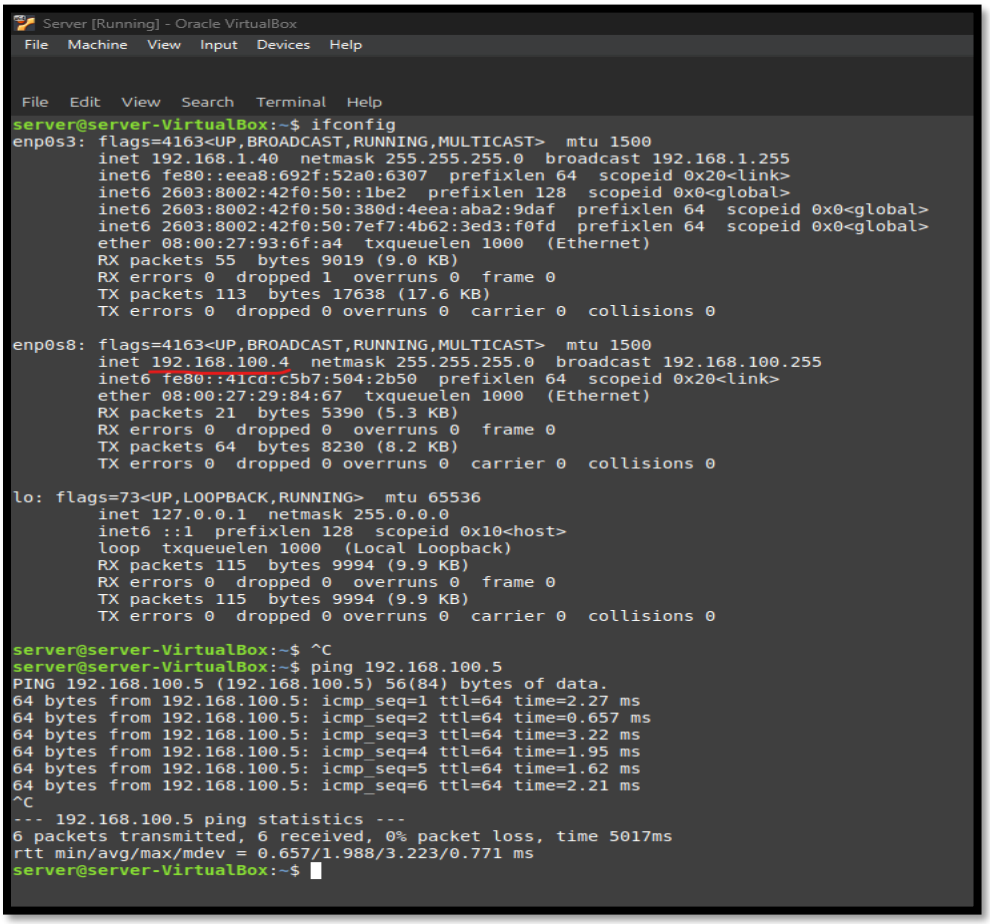

Then, I created one adapter for server and client (and attached it to the NAT network as above).




Server [Running] - Oracle VirtualBox
File Machine View Input Devices Help
server@server-VirtualBox: ~
File Edit View Search Terminal Help
server@server-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.40 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::eea8:692f:52a0:6307 prefixlen 64 scopeid 0x20<link>
inet6 2603:8002:42f0:50::1be2 prefixlen 128 scopeid 0x0<global>
inet6 2603:8002:42f0:50:380d:4eea:aba2:9daf prefixlen 64 scopeid 0x0<global>
inet6 2603:8002:42f0:50:7ef7:4b62:3ed3:f0fd prefixlen 64 scopeid 0x0<global>
ether 08:00:27:93:6f:a4 txqueuelen 1000 (Ethernet)
RX packets 55 bytes 9019 (9.0 KB)
RX errors 0 dropped 1 overruns 0 frame 0
TX packets 113 bytes 17638 (17.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.100.4 netmask 255.255.255.0 broadcast 192.168.100
inet6 fe80::41cd:c5b7:504:2b50 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:29:84:67 txqueuelen 1000 (Ethernet)
RX packets 21 bytes 5390 (5.3 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 64 bytes 8230 (8.2 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 115 bytes 9994 (9.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 115 bytes 9994 (9.9 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
server@server-VirtualBox:~$ ^C
server@server-VirtualBox:~$ ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
64 bytes from 192.168.100.5: icmp_seq=1 ttl=64 time=2.27 ms
64 bytes from 192.168.100.5: icmp_seq=2 ttl=64 time=0.657 ms
64 bytes from 192.168.100.5: icmp_seq=3 ttl=64 time=3.22 ms
64 bytes from 192.168.100.5: icmp_seq=4 ttl=64 time=1.95 ms
64 bytes from 192.168.100.5: icmp_seq=5 ttl=64 time=1.62 ms
64 bytes from 192.168.100.5: icmp_seq=6 ttl=64 time=2.21 ms
^C
--- 192.168.100.5 ping statistics ---
Server - Settings
Basic
Expert
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
NAT Network
Name:
KaerMorhen_Network
Adapter Type:
Promiscuous Mode:
Deny
MAC Address:
080027298467
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4

Next step was to make sure the server and the client are communicating. For this, I figured out the IP addresses of the machines and made them ping using the ID addresses. The bottom screenshots show them communicating.

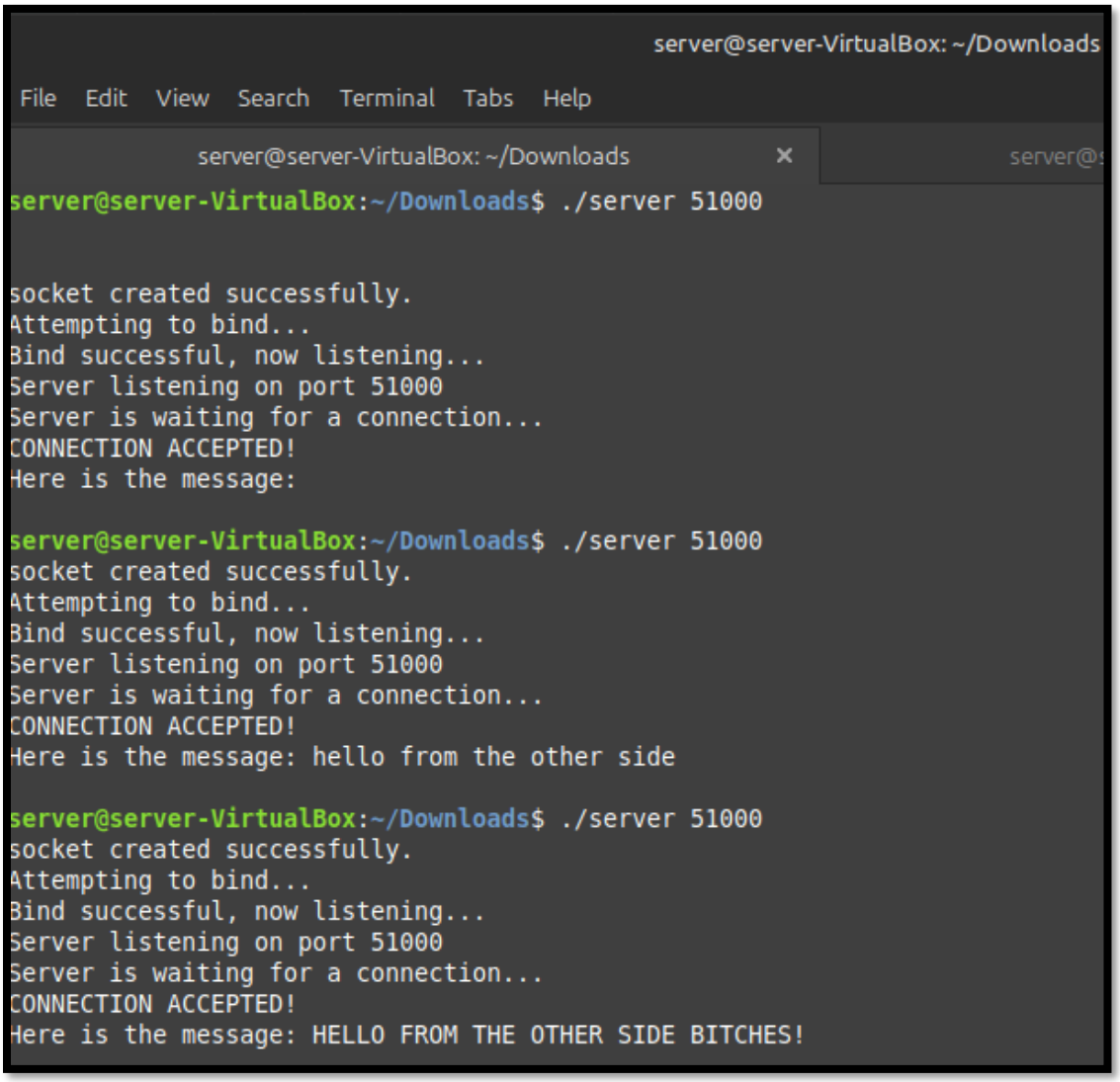


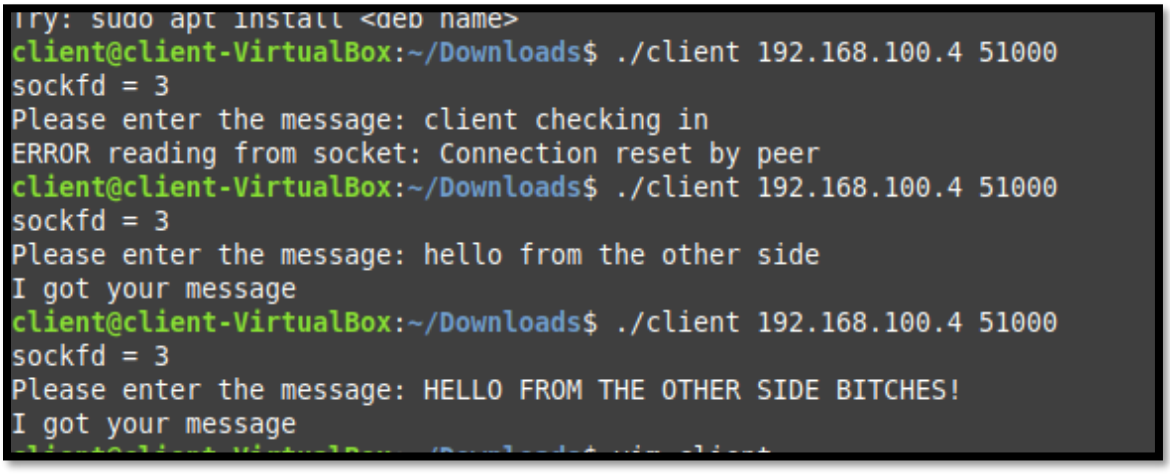



**PART 3:**

Next step was to actually run the executables, run the server and wait for it to listen after the binding to the address gets done.

Then we run the client, which basically makes the server go, “CONNECTION ACCEPTED!”. Finally, we send a message to the server by typing it in the client VM and see it live on the server (as shown in the images below).

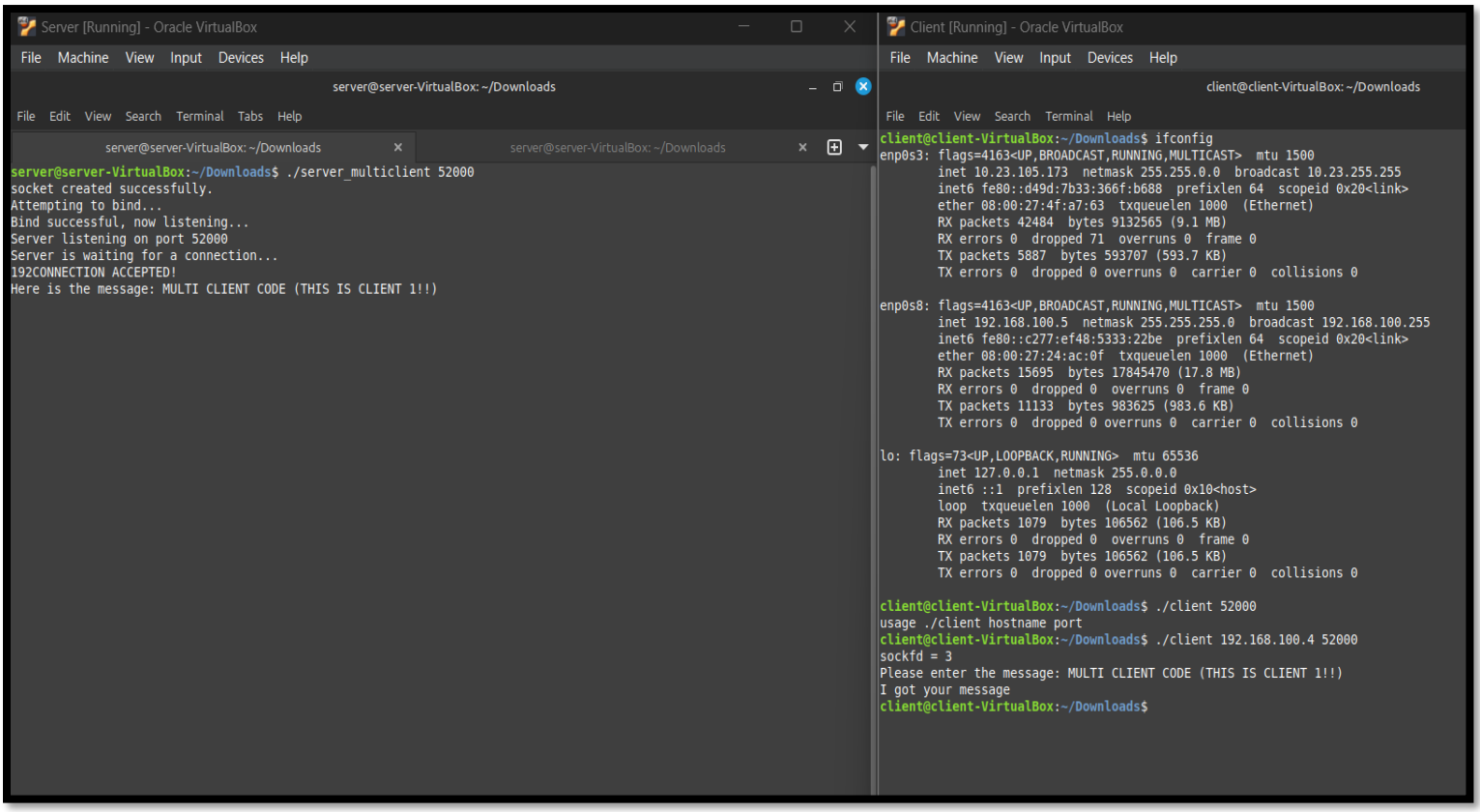


**PART 4**

For the multi-client connections, I made changes to the code as asked by the professor in the document and some of my own. Ran the executables as above and see the output. Only this time, the server didn't drop the connection after receiving message from the client. It was waiting for some other connection to take place (evident by the blinking cursor)

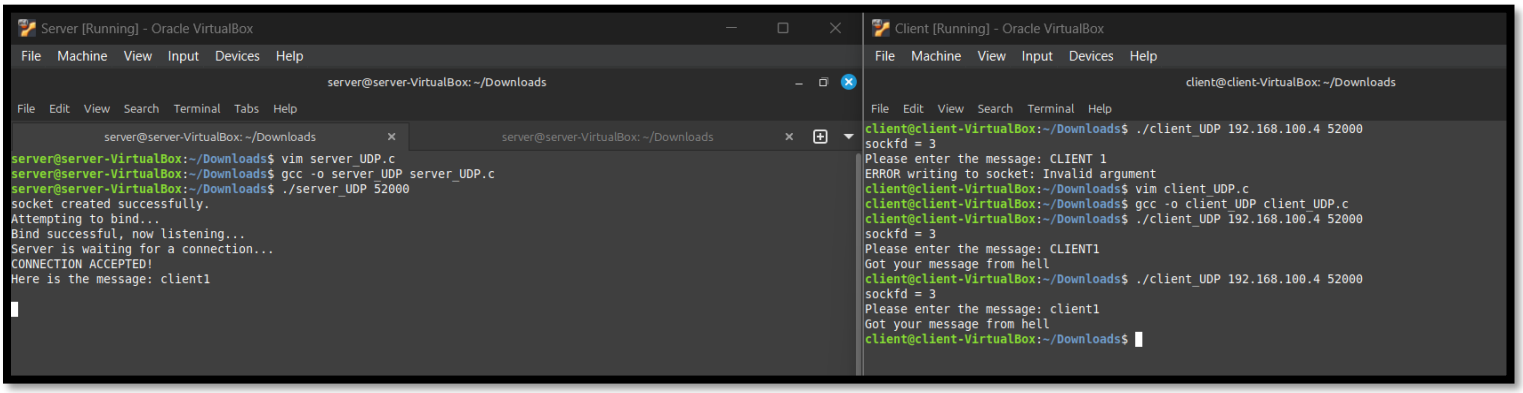
I did not make any changes to the client-side code for this part.

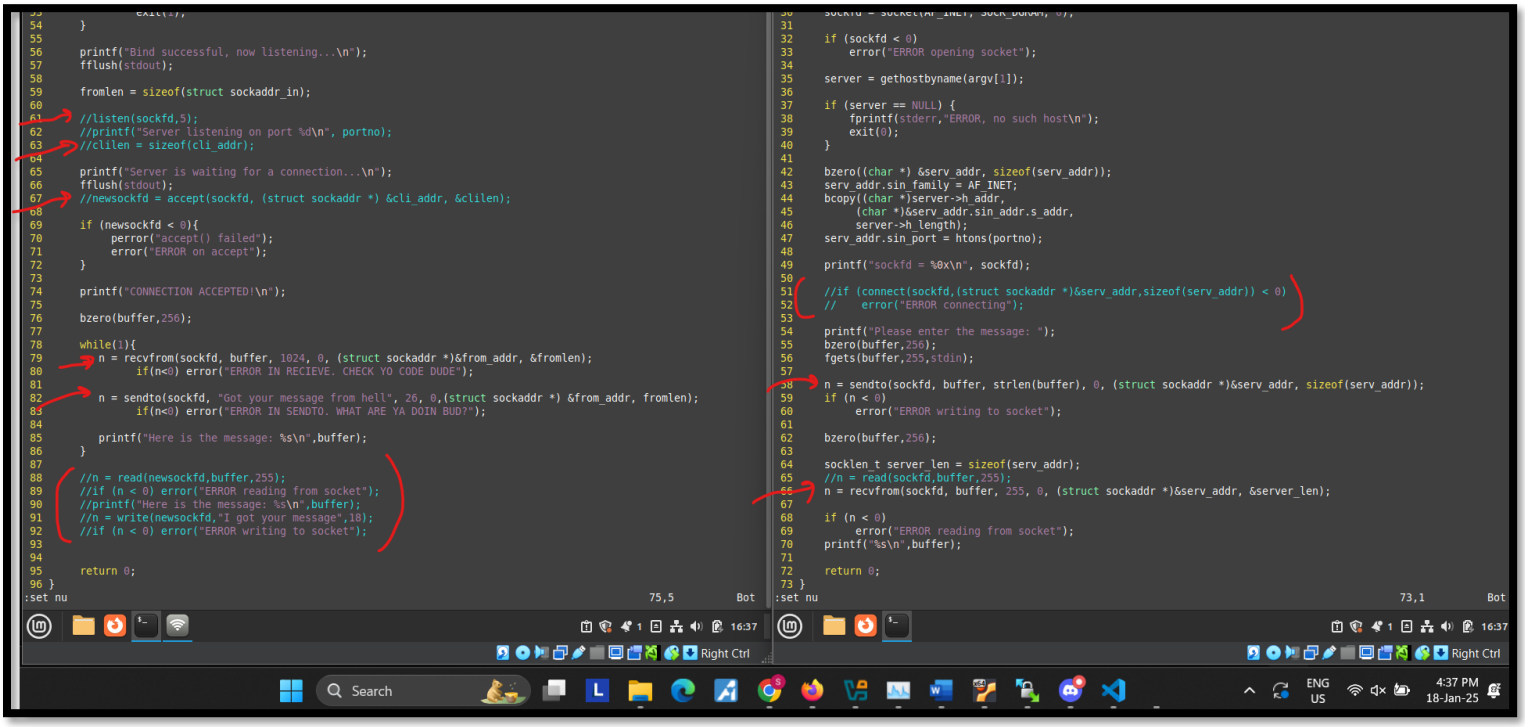


**PART 5**

Finally, we needed to run and play around with datagram sockets and UDP connections.

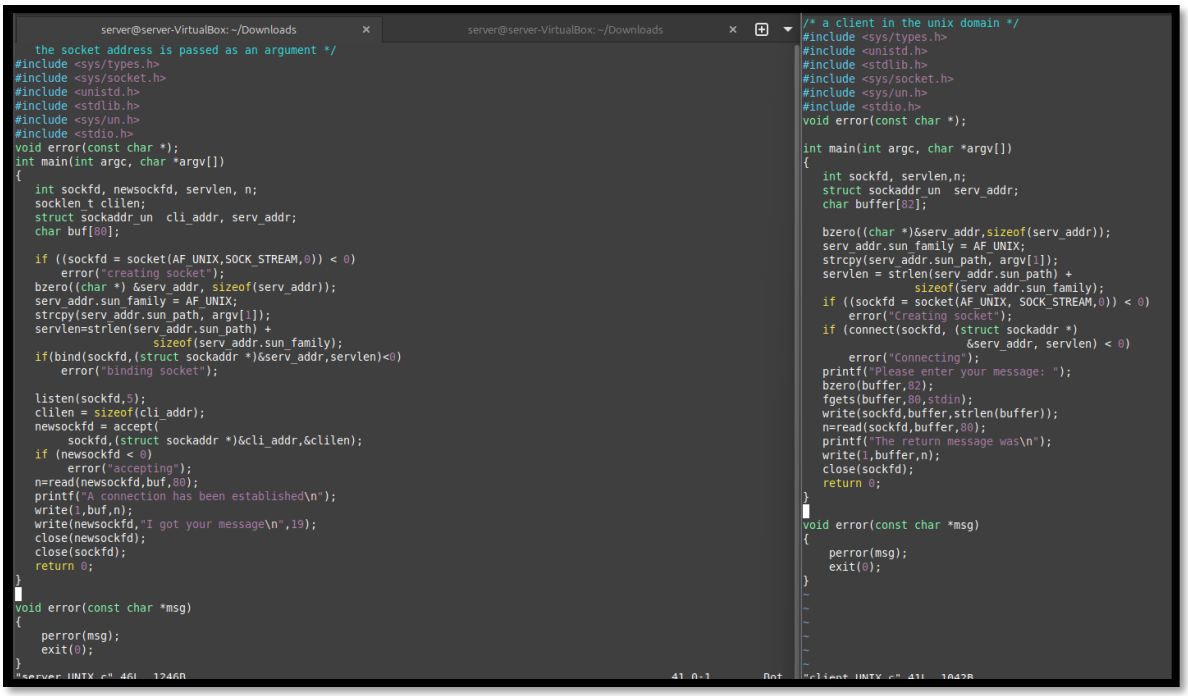
For this we made wholesale changes to the code. After compiling, we ran the code and were FINALLY able to communicate using UDP sockets. See the images below.



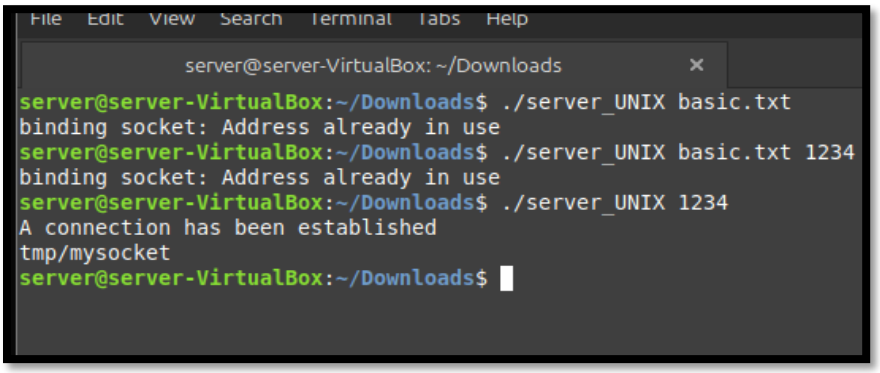
**PART 6**

I tried working on the UNIX based server-client system, and I opened two different tabs of the same VM. Ran the server code and then the client code in a different tab(SAME VM) and got the socket to accept connection and respond



```
server@server-VirtualBox: ~/Downloads
   the socket address is passed as an argument */
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/un.h>
#include <stdio.h>
void error(const char *);
int main(int argc, char *argv[])
{
   int sockfd, newsockfd, servlen, n;
   socklen_t clilen;
   struct sockaddr_un  cli_addr, serv_addr;
   char buf[80];

   if ((sockfd = socket(AF_UNIX,SOCK_STREAM,0)) < 0)
       error("creating socket");
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sun_family = AF_UNIX;
   strcpy(serv_addr.sun_path, argv[1]);
   servlen=strlen(serv_addr.sun_path) +
                   sizeof(serv_addr.sun_family);
   if(bind(sockfd,(struct sockaddr *)&serv_addr,servlen)<0)
       error("binding socket");

   listen(sockfd,5);
   clilen = sizeof(cli_addr);
   newsockfd = accept(
        sockfd,(struct sockaddr *)&cli_addr,&clilen);
   if (newsockfd < 0)
        error("accepting");
   n=read(newsockfd,buf,80);
   printf("A connection has been established\n");
   write(1,buf,n);
   write(newsockfd,"I got your message\n",19);
   close(newsockfd);
   close(sockfd);
   return 0;
}

void error(const char *msg)
{
    perror(msg);
    exit(0);
}
```

```
server@server-VirtualBox: ~/Downloads
/* a client in the unix domain */
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>
void error(const char *);

int main(int argc, char *argv[])
{
   int sockfd, servlen,n;
   struct sockaddr_un  serv_addr;
   char buffer[82];

   bzero((char *)&serv_addr,sizeof(serv_addr));
   serv_addr.sun_family = AF_UNIX;
   strcpy(serv_addr.sun_path, argv[1]);
   servlen = strlen(serv_addr.sun_path) +
                   sizeof(serv_addr.sun_family);
   if ((sockfd = socket(AF_UNIX, SOCK_STREAM,0)) < 0)
       error("Creating socket");
   if (connect(sockfd, (struct sockaddr *)
                       &serv_addr, servlen) < 0)
       error("Connecting");
   printf("Please enter your message: ");
   bzero(buffer,82);
   fgets(buffer,80,stdin);
   write(sockfd,buffer,strlen(buffer));
   n=read(sockfd,buffer,80);
   printf("The return message was\n");
   write(1,buffer,n);
   close(sockfd);
   return 0;
}

void error(const char *msg)
{
    perror(msg);
    exit(0);
}
```



```
File  Edit  View  Search  Terminal  Tabs  Help
server@server-VirtualBox: ~/Downloads
server@server-VirtualBox:~/Downloads$ ./server_UNIX basic.txt
binding socket: Address already in use
server@server-VirtualBox:~/Downloads$ ./server_UNIX basic.txt 1234
binding socket: Address already in use
server@server-VirtualBox:~/Downloads$ ./server_UNIX 1234
A connection has been established
tmp/mysocket
server@server-VirtualBox:~/Downloads$
```

```
server@server-VirtualBox:~/Downloads$ cp client.c client_UNIX.c
server@server-VirtualBox:~/Downloads$ vim client_UNIX.c
server@server-VirtualBox:~/Downloads$ gcc -o client_UNIX client_UNIX.c
server@server-VirtualBox:~/Downloads$ ./client_UNIX 1234
Please enter your message: tmp/mysocket
The return message was
I got your message
server@server-VirtualBox:~/Downloads$
```

***GITHUB LINK FOR MY REPOSITORY:***

You'll find all the codes and the executables on this repository.

https://github.com/SARTHAK-JAIN-ASIC/EE533/tree/main/LAB1