Name: Neeraj Varma PRN: 1032210651
Roll No: 08
Batch: A1

__/__/____

CN Lab Assignment

Theory:

(i) Client/Server Communication: Client & Servers exchange message in a request response messaging pattern. The client sends a request & the server returns a response. This exchange is an example of inter-process communication

(ii) Introduction to TCP: TCP is a standard that defines how to establish & maintain a network conversation by which applications can exchange data TCP works with the IP, which defines how computers send packets of data to each other

(iii) TCP segment header: It is a 4-bit field that indicates the length of the TCP header by a no. of 4-byte words in the header

(iv) TCP connection establishment & release: TCP uses a three-way handshake to establish a reliable connection. The exchange of these flags is performed in three steps; SYN, SYN-ACK, ACK

(v) Socket: A process sends messages into & receives messages from the network through a software interface called a socket

(vi) TCP Socket functions: It is able to listen on the TCP port for incoming connections. The TCP socket is able to initiate a connection to remote server.

__/__/____

(vii) TCP Socket Flow:

Server:	Client:
Socket	Socket()
↓	↓
Bind	Connect()
↓	↓
Listen	Write()
↓	↓
Accept	Read()
↓	↓
Send/Rec	Close()
↓	
Exit	

FAQ's

1] State the IANA range for ports list at least well known ports

→ Well known ports: 0 - 1023

Registered ports: 1024 - 49151

Dynamic ports: 49152 - 65535

Some well known ports:

① 7 - Echo

② 20, 21, - FTP

③ 37 - Time

④ 53 - DNS

⑤ 80 - HTTP

__/__/____

2] If bind() fails, what should one do with socket descriptor?

→ The unix system will close all open file descriptors on exit. If the code is not exited, the programmer can close it with close()

3] Draw & explain header

→

Source port	Destination port		
Sequence number			
Acknowledgement number			
Header Length	RSV	Flags	Window
		Urgent pointer	
Options			

① Source port: Specifies sender
② Destination port: Specifies receiver
③ Acknowledgment: Used by receiver to request next TCP segment
④ Window: Specifies how many bytes the receiver is willing to receive

Code [C]:

Chat Client :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<errno.h>
#include<fcntl.h> void
main(){
 struct sockaddr_in server,client; int
sock,clientSocket; char
receivedBytes[1024],sendBytes[1024]; int
bytes; if((sock =
socket(AF_INET,SOCK_STREAM,0)) == - 1){
perror("Invalid Socket Descriptor"); exit(1);
}
 server.sin_family = AF_INET;
server.sin_port = htons(5005);
```

```
server.sin_addr.s_addr = INADDR_ANY;
bzero(&(server.sin_zero),8);

if(connect(sock,(struct sockaddr
*)&server,sizeof(server)) == - 1){
perror("Unable to connect");
exit(1);
}  while(1){
printf("\nClient: ");
gets(sendBytes);
bytes = send(sock,sendBytes,1024,0);
if(strcmp(sendBytes,"q")
== 0 || strcmp(sendBytes,"Q") == 0){
printf("\nClient exiting...");
close(sock); exit(1); } bytes =
recv(sock,receivedBytes,1024,0);
receivedBytes[bytes] = '\0';
printf("\nServer: %s",receivedBytes);
if(strcmp(receivedBytes,"q") == 0 ||
strcmp(receivedBytes,"Q") == 0){
printf("\nServer going off...");
close(sock); break; }

}

}
```

Chat Server :

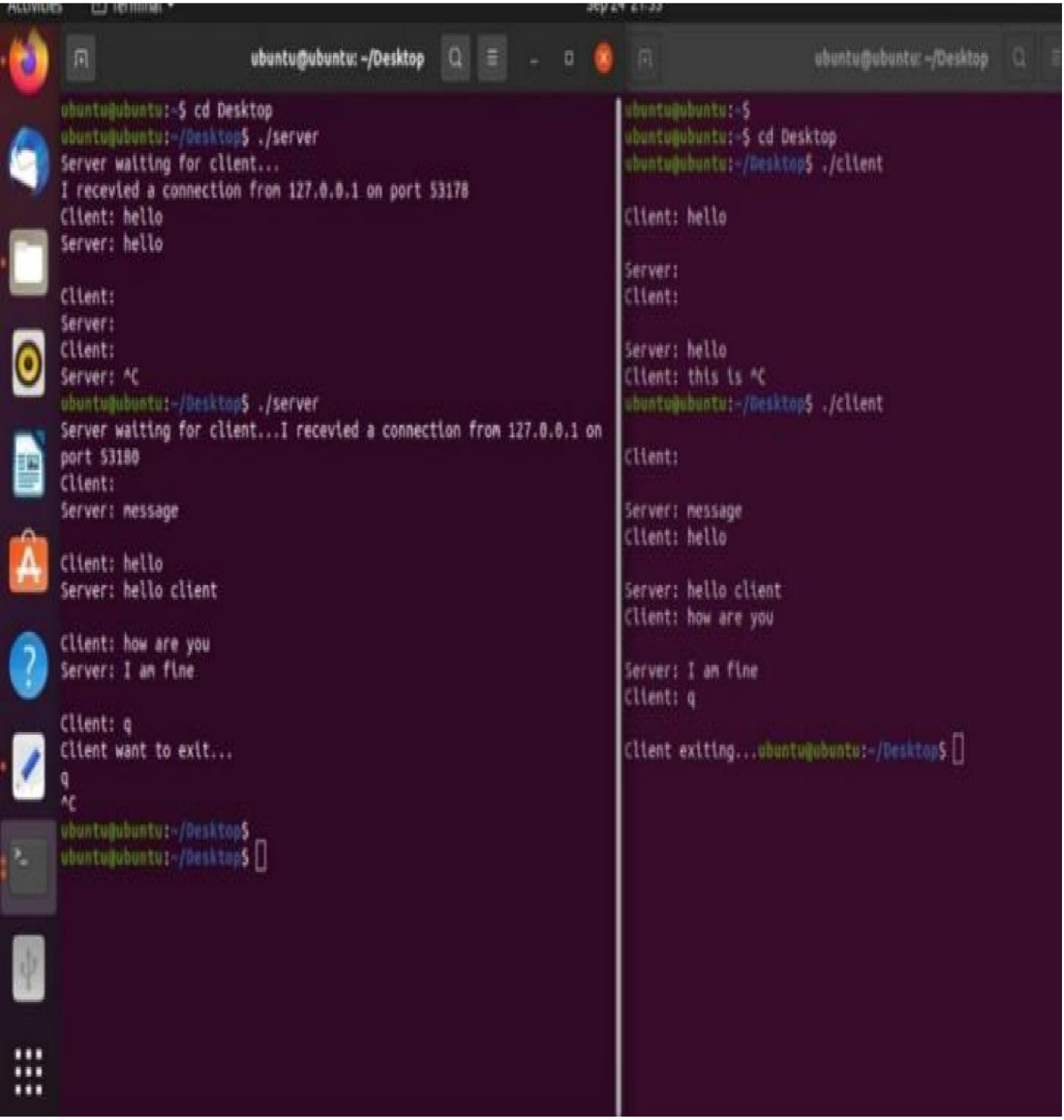
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h> #include<arpa/inet.h>

#include<errno.h> #include<fcntl.h>
 void
main(){
```

```
 struct sockaddr_in server,client; int
sock,clientSocket; char
receivedBytes[1024],sendBytes[1024]; int
bytes; if((sock =
socket(AF_INET,SOCK_STREAM,0)) == - 1){
perror("Invalid Socket Descriptor"); exit(1);
}

server.sin_family = AF_INET;
server.sin_port = htons(5005);
server.sin_addr.s_addr = INADDR_ANY;
bzero(&(server.sin_zero),8);

if(bind(sock,(struct sockaddr
*)&server,sizeof(server)) == - 1){
perror("Unable to bind"); exit(1);
}  if(listen(sock,5) == -1){
perror("Unable to listen"); exit(1); }
printf("Server waiting for client...");
 fflush(stdout);
 while(1){ int len = sizeof(client);
clientSocket = accept(sock,(struct sockaddr
*)&client,&len); if(clientSocket == -1){
perror("Connection error");
```

```
exit(1); } printf("I recevied a connection
from %s on
port
%d",inet_ntoa(client.sin_addr),ntohs(client.sin_por
t));  while(1){ bytes =
recv(clientSocket,receivedBytes,1024,0);
receivedBytes[bytes] = '\0'; printf("\nClient:
%s",receivedBytes);
if(strcmp(receivedBytes,"q") == 0 ||
strcmp(receivedBytes,"Q") ==
0){ printf("\nClient want to exit...");
printf("\nWaiting for new client...");
close(clientSocket); break; }
printf("\nServer: "); gets(sendBytes); bytes
= send(clientSocket,sendBytes,1024,0);
if(strcmp(sendBytes,"q") == 0 ||
strcmp(sendBytes,"Q") == 0){
printf("\nServer going off...");
close(clientSocket);
exit(1);
```

}}

Output


ubuntu@ubuntu: ~/Desktop
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ ./server
Server waiting for client...
I recevied a connection from 127.0.0.1 on port 53178
Client: hello
Server: hello
Client:
Server:
Client:
Server: ^C
ubuntu@ubuntu:~/Desktop$ ./server
Server waiting for client...I recevied a connection from 127.0.0.1 on port 53180
Client:
Server: message
Client: hello
Server: hello client
Client: how are you
Server: I am fine
Client: q
Client want to exit...
q
^C
ubuntu@ubuntu:~/Desktop$
ubuntu@ubuntu:~/Desktop$
ubuntu@ubuntu: ~/Desktop
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ ./client
Client: hello
Server:
Client:
Server: hello
Client: this is ^C
ubuntu@ubuntu:~/Desktop$ ./client
Client:
Server: message
Client: hello
Server: hello client
Client: how are you
Server: I am fine
Client: q
Client exiting...ubuntu@ubuntu:~/Desktop$