Computer Networks Lab
Week-6
Aug 30, 2021

Venkata Naga Sai Ram Nomula
RA1911033010021
L2 - SWE

GIVEN REQUIREMENTS: There are two hosts, Client and Server. Both the Client and the Server exchange messages i.e. they send messages or receive messages from the other. There is only a single way communication between them.

TECHNICAL OBJECTIVE: To implement a half duplex application, where the Client establishes a connection with the Server. The Client can send and the server will receive messages at the same time.

METHODOLOGY:

Server:

- Include the necessary header files.
- Create a socket using the socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET, sin_addr to INADDR_ANY, sin_port to dynamically
- assigned port number.
- Bind the local host address to the socket using the bind function.
- Listen on the socket for connection requests from the client.
- Accept connection requests from the Client using the accept function.
- Fork the process to receive a message from the client and print it on the console.
- Read a message from the console and send it to the client.

Client:

- Include the necessary header files.

- Create a socket using the socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET.
- Get the server IP address and the Port number from the console.
- Using gethostbyname function assign it to a hostent structure, and assign it to sin_addr of
- the server address structure.
- Request a connection from the server using the connect function.
- Fork the process to receive a message from the server and print it on the console.
- Read a message from the console and send it to the server.

CODE:

Server: server.c

```
#include<sys/types.h>
#include<stdio.h>
#include<string.h>
#include<netdb.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<netinet/in.h>
int main(int argc,char *argv[])
{
int n,sd,ad;
struct sockaddr_in servaddr,cliaddr;
socklen_t clilen,servlen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(5000);
/*TCP socket is created, an Internet socket address structure is filled with
```

```
wildcard address & server's well known port*/
sd=socket(AF_INET,SOCK_STREAM,0);
/*Bind function assigns a local protocol address to the socket*/
bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
/*Listen function specifies the maximum number of connections that kernel
should queue for this socket*/
listen(sd,5);
printf("%s\n","server is running...");
/*The server to return the next completed connection from the front of the
completed connection Queue calls it*/
ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
while(1)
{
bzero(&buff,sizeof(buff));
/*Receiving the request from client*/
recv(ad,buff,sizeof(buff),0);
printf("Receive from the client:%s\n",buff);
n=1;
while(n==1)
{
bzero(&buff1,sizeof(buff1));
printf("%s\n","Enter the input data:");
/*Read the message from client*/
fgets(buff1,10000,stdin);
/*Sends the message to client*/
send(ad,buff1,strlen(buff1)+1,0);
printf("%s\n","Data sent");
n=n+1;
}
}
return 0;
}
```

Client: client.c

```
#include<sys/types.h>
```

```
#include<sys/socket.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdio.h>
#include<string.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
int n,sd,cd;
struct sockaddr_in servaddr,cliaddr;
socklen_t servlen,clilen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(5000);
/*Creating a socket, assigning IP address and port number for that socket*/
sd=socket(AF_INET,SOCK_STREAM,0);
/*Connect establishes connection with the server using server IP address*/
cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
while(1)
{
bzero(&buff,sizeof(buff));
printf("%s\n","Enter the input data:");
/*This function is used to read from server*/
fgets(buff,10000,stdin);
/*Send the message to server*/
send(sd,buff,strlen(buff)+1,0);
printf("%s\n","Data sent");
n=1;
while(n==1)
{
bzero(&buff1,sizeof(buff1));
```

```
/*Receive the message from server*/
recv(sd,buff1,sizeof(buff1),0);
printf("Received from the server:%s\n",buff1);
n=n+1;
}
}
return 0;
}
```

Result:

server.c

```
{
int n,sd,ad;
struct sockaddr_in servaddr,cliaddr;
socklen_t clilen,servlen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(6000);
/*TCP socket is created, an Internet socket address structure is fillec
wildcard address & server's well known port*/
sd=socket(AF_INET,SOCK_STREAM,0);
/*Bind function assigns a local protocol address to the socket*/
bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
/*Listen function specifies the maximum number of connections that kerr
should queue for this socket*/
listen(sd,5);
printf("%s\n","server is running...");
/*The server to return the next completed connection from the front of
completed connection Queue calls it*/
ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
while(1)
{
bzero(&buff,sizeof(buff));
/*Receiving the request from client*/
recv(ad,buff,sizeof(buff),0);
printf("Receive from the client:%s\n",buff);
n=1;
while(n==1)
{
bzero(&buff1,sizeof(buff1));
printf("%s\n","Enter the input data:");
/*Read the message from client*/
```

19:31 C and C++ Spaces: 4

client.c

```
#include<unistd.h>
#include<stdio.h>
#include<string.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
int n,sd,cd;
struct sockaddr_in servaddr,cliaddr;
socklen_t servlen,clilen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(6000);
/*Creating a socket, assigning IP address and port number for that sock
sd=socket(AF_INET,SOCK_STREAM,0);
/*Connect establishes connection with the server using server IP addres
cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
while(1)
{
bzero(&buff,sizeof(buff));
printf("%s\n","Enter the input data:");
/*This function is used to read from server*/
fgets(buff,10000,stdin);
/*Send the message to server*/
send(sd,buff,strlen(buff)+1,0);
printf("%s\n","Data sent");
n=1;
while(n==1)
{
bzero(&buff1,sizeof(buff1));
/*Receive the message from server*/
recv(sd,buff1,sizeof(buff1),0);
printf("Received from the server:%s\n",buff1);
```

19:31 C and C++ Spaces: 4

./a.out - "ip-172-31-9-200"

```
RA1911033010025:~/environment $ cd RA1911033010021
RA1911033010025:~/environment/RA1911033010021 $ cd 'Half Duplex'
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ cc server.c
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ ./a.out
Segmentation fault (core dumped)
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ ./a.out
Segmentation fault (core dumped)
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ cc server.c
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ ./a.out
server is running...
Receive from the client:Hello!

Enter the input data:
Hi!
Data sent
```

./a.out - "ip-172-31-9-200"

```
RA1911033010025:~/environment $ cd RA1911033010021
RA1911033010025:~/environment/RA1911033010021 $ cd 'Half Duplex'
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ cc client.c
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ cc client.c
RA1911033010025:~/environment/RA1911033010021/Half Duplex $ ./a.out 127.0.0.1
Enter the input data:
Hello!
Data sent
Received from the server:Hi!

Enter the input data:
```