DS Assignment 4

Name: Himani Verma

Admission No: U19CS075

1. Extend your echo Client Server message passing application to chat application.

● Client and Server are able to send the message to each other until one of them quits or terminates.

Source Code:

Server Side:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

void func(int connfd)
{
	char buff[MAX];
	int n;
	// infinite loop for chat
	for (;;) {
		bzero(buff, MAX);
		// read the message from client and copy it in buffer
		read(connfd, buff, sizeof(buff));
		// print buffer which contains the client contents
```

```
                printf("From client: %s\t To client : ", buff);
                bzero(buff, MAX);
                n = 0;
                // copy server message in the buffer
                while ((buff[n++] = getchar()) != '\n');

                // and send that buffer to client
                write(connfd, buff, sizeof(buff));

                if (strncmp("exit", buff, 4) == 0) {
                        printf("Server Exit...\n");
                        break;
                }
        }
}

int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf("socket creation failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));

        // assign IP, PORT
```

```
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servaddr.sin_port = htons(PORT);

	// Binding newly created socket to given IP and verification
	if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
		printf("socket bind failed...\n");
		exit(0);
	}
	else
		printf("Socket successfully binded..\n");

	// Now server is ready to listen and verification
	if ((listen(sockfd, 5)) != 0) {
		printf("Listen failed...\n");
		exit(0);
	}
	else
		printf("Server listening..\n");
	len = sizeof(cli);
	connfd = accept(sockfd, (SA*)&cli, &len); // Accept the data packet from client and
verification
	if (connfd < 0) {
		printf("server accept failed...\n");
		exit(0);
	}
	else
		printf("server accept the client...\n");

	func(connfd); // Function for chatting between client and server
```

```
	close(sockfd); // After chatting close the socket
}
```

Client Side:

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
	char buff[MAX];
	int n;
	for (;;) {
		bzero(buff, sizeof(buff));
		printf("Enter the string : ");
		n = 0;
		while ((buff[n++] = getchar()) != '\n')
			;
		write(sockfd, buff, sizeof(buff));
		bzero(buff, sizeof(buff));
		read(sockfd, buff, sizeof(buff));
		printf("From Server : %s", buff);
		if ((strncmp(buff, "exit", 4)) == 0) {
			printf("Client Exit...\n");
			break;
		}
	}
```

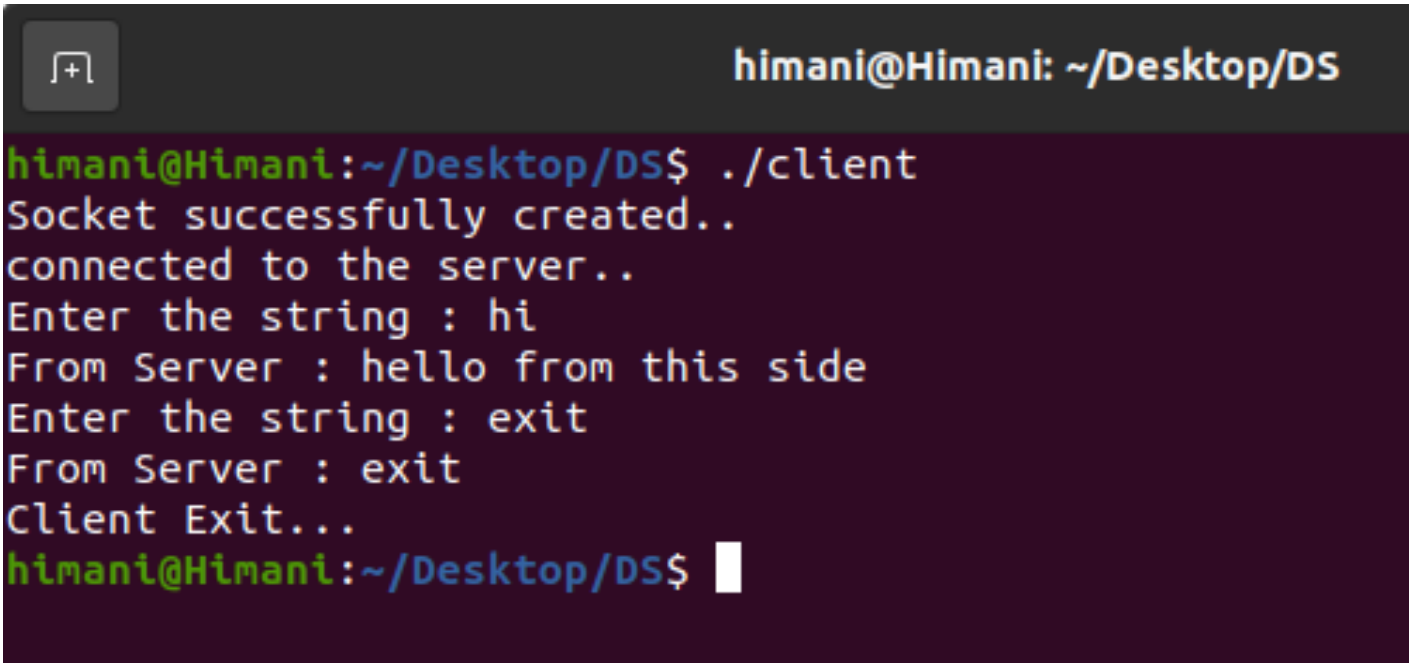
```
}

int main()
{
	int sockfd, connfd;
	struct sockaddr_in servaddr, cli;
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd == -1) {
		printf("socket creation failed...\n");
		exit(0);
	}
	else
		printf("Socket successfully created..\n");
	bzero(&servaddr, sizeof(servaddr));

	// assign IP, PORT
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
	servaddr.sin_port = htons(PORT);

	// connect the client socket to server socket
	if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
		printf("connection with the server failed...\n");
		exit(0);
	}
	else
		printf("connected to the server..\n");
	func(sockfd); // function for chat
	close(sockfd);
}
```

Output:

Server Side:

```
himani@Himani:~/Desktop/DS$ ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server accept the client...
From client: hi
         To client : hello from this side
From client: exit
         To client : exit
Server Exit...
himani@Himani:~/Desktop/DS$
```

Client Side:



2. Using the Client-Server communication mechanism get the load status of other nodes in your network (identify the states of other nodes in the system – Overload, Moderate, Lightly).

- Implement the Client-Server model. Run the client and server instance on same machine and pass the message from client to server or server to client
- Get the CPU load of the client or server and state that either it is under loaded or overloaded.

The client server communication mechanism has the limitation that it only handles one

connection at a time and then terminates. A real-world server should run indefinitely and should have the capability of handling a number of simultaneous connections, each in its own process.

Source Code:

Server Side:

```
#include<stdio.h>
#include<netinet/in.h>
#include<netdb.h>
#include<sys/types.h>
#include<sys/stat.h>
#include <unistd.h>
#define SERV_TCP_PORT 5035

int main(int argc,char**argv)
{
        int sockfd, newsockfd, clength;
        struct sockaddr_in serv_addr, cli_addr;
        char buffer[4096];

        //Create socket
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        puts("Socket created");

        //Prepare the sockaddr_in structure
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(SERV_TCP_PORT);

        //Bind
        printf("\nStart");
```

```
        bind(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
        printf("\nListening...");
        printf("\n");
        listen(sockfd, 5);

        //accept connection from an incoming client
        clength = sizeof(cli_addr);
        newsockfd = accept(sockfd, (struct sockaddr*)&cli_addr, &clength);
        printf("\nAccepted");
        printf("\n");

        //print message from client
        read(newsockfd, buffer, 4096);
        printf("\nClient message:%s", buffer);
        write(newsockfd,buffer, 4096);
        printf("\n");

        //close the socket
        close(sockfd);
        return 0;
}
```

Client Side:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include <arpa/inet.h>
#include<netinet/in.h>
#include<netdb.h>
#include <unistd.h>
#define SERV_TCP_PORT 5035
```

```
int main(int argc,char*argv[])
{
	int sockfd;
	struct sockaddr_in serv_addr;
	struct hostent *server;
	char buffer[4096];

	//Create socket
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
	serv_addr.sin_port = htons(SERV_TCP_PORT);

	//Connect to remote server
	printf("\nReady for sending...");
	connect(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
	printf("\nClient Information: ");

	FILE *pp;
	pp = popen("./Ass2.sh", "r");
	if (pp != NULL) {
		while (1) {
			char *line;
			line = fgets(buffer, sizeof buffer, pp);
			if (line == NULL) break;
				printf("%s", line); /* line includes '\n' */
		}
		pclose(pp);
	}
```

```
        //send to server
        write(sockfd,buffer,4096);
        printf("Serverecho:%s",buffer);
        printf("\n");
        close(sockfd);
        return 0;
}
```

Output:

Server Side:

```
himani@Himani:~/Desktop/DS$ ./a2
Socket created

Start
Listening...

Accepted

Client message:Lightly Loaded

himani@Himani:~/Desktop/DS$
```

Client Side:

```
himani@Himani:~/Desktop/DS$ gcc loadclient.c -o a1
himani@Himani:~/Desktop/DS$ gcc loadserver.c -o a2
himani@Himani:~/Desktop/DS$ ./a1

Ready for sending...
Client Information: CPU Usage: 2.7%
Lightly Loaded
Serverecho:Lightly Loaded
```