1911

Server code:

```
/*
	1911
	Server code for the TCP client-server model.
*/



#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <stdlib.h> //for exit
#include <string.h>
#include <arpa/inet.h>

#define LISTEN_BACKLOG 5

void server_work(int new_sock){

	/*
		function to do the server work.
		echo the client's input back to the client's screen.
		Also print the client's message on the server terminal.
	*/

	char msg[50];

	char buf[10];

	read(new_sock, buf, 10); //read the message from the clent socket

	bzero(msg, sizeof(msg) );

	strcpy(msg, "\n\nClient\'s message: ");

	write(1, msg, sizeof(msg) );

	write(1, buf, sizeof(buf) );

	//write(1, "\n", 1 );

	//bzero(msg, sizeof(msg) );

	//strcpy(msg, "Server Response: Message received.\n");

	//write(new_sock, msg, sizeof(msg) );

	write(new_sock, buf, sizeof(buf) ); //write the client message back to the client terminal
```

```
}

void main(){

        struct sockaddr_in myaddr, client_addr;

        int sock_id;

        bzero(&myaddr, sizeof(myaddr) );

        sock_id = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); //create a socket


        //initialize the server address and port number

        myaddr.sin_family = AF_INET;

        myaddr.sin_port = htons(9011);

        myaddr.sin_addr.s_addr = inet_addr("127.0.0.1"); //htonl('127.0.0.1')

        //if theere is any error while creating the socket

        if (sock_id == -1){
                write(1, "socket error", 12);
                exit(-1);
        }

        //write(1, "socket created", 14);

        int len = sizeof(myaddr);

        /*      bind the socket created to the server address
                if error, display error message

        */

        if ( bind(sock_id, (struct sockaddr*)&myaddr, len) == -1 ){
                write(1, "bind error", 10);
                exit(-1);

        }

        //write(1, "bind created", 12);

        /*      then the listen process; convert the active socket to passive
                if error, display error message

        */

        if ( listen(sock_id, LISTEN_BACKLOG) == -1 ){
                write(1, "listen error", 12);
```

```
                exit(-1);

        }

        // write(1, "listen created", 14);

        int new_sock, child_pid;
        int client;

        for(;;){

                client = sizeof(client_addr);

                // create new socket to accept client's requests using accept
                new_sock = accept(sock_id, (struct sockaddr *)&client_addr, &client);


                // if there is error in accept method

                if ( new_sock == -1 ){
                        write(1, "accept error", 12);
                        exit(-1);
                }

                //write(1, "new_sock created", 16);

                /*      create child process for handling the client's requests using
fork()
                        when fork returns 0; it will be in the child process; and child
process will be exected.

                */

                if ( (child_pid = fork() ) == 0 ){

                        server_work(new_sock); // call the method to to the server work


                        close(new_sock); // close the new socket created after the server
work is done
                        exit(0);
                }

                close(new_sock); // close the new socket in the parent proess

        }
}
```

Client code:

```
/*
		1911
		Client code for the TCP client-server model
*/

#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <stdlib.h> //for exit
#include <string.h>
#include <arpa/inet.h>

void main(){

		struct sockaddr_in client_addr;

		int sock_id;

		bzero(&client_addr, sizeof(client_addr) );

		// create socket for the client
		sock_id = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

		// initialize the client address and port number

		client_addr.sin_family = AF_INET;
		client_addr.sin_port = htons(9011);
		client_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

		// if there is error in creating the socket
		if (sock_id == -1){
				write(1, "socket error", 12);
				exit(-1);
		}

		int len = sizeof(client_addr);

		/*		connect call, to connect to ther server
				if error display error message
		*/

		if ( connect(sock_id, (struct sockaddr*)&client_addr, len) == -1 ){
				write(1, "connect error.", 14);
				exit(-1);
		}

		/*
				read a ten character input from the client,
				and send it to the server.
				Write to the terminal, the response from the server.
```

```
        */

        char buff[10];

        char msg[60] = "Client Online: Enter message of ten characters only: ";
        write(1, msg, sizeof(msg) );

        read(0, buff, 10);

        write(sock_id, buff, sizeof(buff) );

        bzero(msg, sizeof(msg) );

        read(sock_id, msg, sizeof(msg));

        write(1, msg, sizeof(msg) );

        write(1, "\n", 1 );

        close(sock_id);

        exit(0);

}
```